Theory of Biaxial Graded-Index Optical Fiber

S.F. Kawalko
Department of Electrical Engineering
and Computer Science
University of Illinois at Chicago

Prepared under grant NAG2-544

Theory of Biaxial Graded-Index Optical Fiber

BY

STEPHEN F. KAWALKO
B.S., University of Illinois at Chicago, 1987

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering in the Graduate College of the
University of Illinois at Chicago, 1990

Chicago, Illinois

Acknowledgement

This research was supported by the Pacific Missile Test Center and the NASA-Ames Research Center under grant NAG2-544.

Table Of Contents

List of Figures

Theory of Biaxial Graded-Index Optical Fiber

1. Introduction

1.1 Review of Previous Research

The optical fiber has become a much studied transmission system due to its property of wave guidance with low loss. In recent years it has been shown that introducing anisotropies into the dielectric medium of the fiber produces several interesting features, such as control of power flow and reduction of peak attenuation near cutoff.

Typically the analysis of wave propagation in a cylindrical dielectric waveguide such as an optical fiber is performed using a wave equation formulation. For the simple case of a step-index fiber a detailed analysis, including dispersion relations, cutoff conditions and mode designations, is presented by Snitzer [1]. Paul and Shevgaonkar [2] present a similar analysis for a uniaxial step-index fiber and also perform a perturbation analysis to determine the modal attenuation constants. These are the only two cases for which exact solutions are known.

For inhomogeneous fibers no exact solutions are known. For the case of an isotropic graded-index fiber several approximate analytic solution methods are available. These approximate solutions all share the common assumption that the fiber is infinite in extent. In addition if the permittivity is assumed to vary slowly over the distance of one wavelength the wave equation formulation simplifies to an asssociated scalar wave equation. If the permittivity profile is parabolic the solution to the scalar wave equation can be written in terms of either Laguerre polynomials [3] if cylindrical coordinates are used or Hermite

polynomials [4] if rectangular coordinates are used. For arbitrary permittivity profiles the scalar wave equation can be solved using the well known WKB solution method [5], [6]. For parabolic permittivity profiles all three solution methods give identical results. Under the assumption that the fields are far from cutoff Kurtz and Streifer [7], [8] have shown that a solution to the full vector problem can be written in terms of either Laguerre polynomials if the permittivity profile is quadratic or asymptotically in terms of Bessel and Airy functions for arbitrary permittivity profiles which decrease slowly and monotonically. A comparison of the vector and scalar solutions for the quadratic permittivity profile implies the vector modes can be obtained by simply renumbering the scalar modes [9]. Using the renumbered scalar modes as a basis Hashimoto [10], [11], [12] and Ikuno [13], [14], [15] have developed two slighly different iterative methods which can be used to solve the full vector problem for an isotropic graded-index fiber.

An alternate formulation of the problem is to write the four first-order differential equations for the tangential field components as a first-order matrix differential equation. For a step-index fiber with uniaxial core and cladding Tonning [16] has shown that the matrix formulation can be solved exactly in terms of Bessel functions. For isotropic graded-index fibers with arbitrary permittivity profiles Yeh and Lingren [17] have indirectly used the matrix fomulation in developing a numerical solution method based on the concept of stratification. Using the concept of transition matrices Tonning [18] has developed a numerical procedure which can be used to solve the matrix differential equation for isotropic graded-index fibers.

1.2 Outline of Proposed Research

This thesis concerns itself with the general case of a biaxial graded-index fiber with a homogeneous cladding. Two methods, wave equation and matrix differential equation, of formulating the problem and their respective solutions will be discussed.

For the wave equation formulation of the problem it will be shown that for the case of a diagonal permittivity tensor, $\bar{\bar{\epsilon}}$, the longitudinal electric and magnetic fields satisfy a pair of coupled second-order differential equations. Also, a generalized dispersion relation is derived in terms of the solutions for the longitudinal electric and magnetic fields. For the case of a step-index fiber, either isotropic or uniaxial, these differential equations can be solved exactly in terms of Bessel functions. For the cases of an isotropic graded-index and a uniaxial graded-index fiber a solution using the Wentzel, Krammers and Brillouin (WKB) approximation technique will be shown. Results for some particular permittivity profiles will be presented. Also the WKB solutions will be compared with the vector solution found by Kurtz and Streifer [7].

For the matrix formulation it will be shown that the tangential components of the electric and magnetic fields satisfy a system of four first-order differential equations which can be conveniently written in matrix form. For the special case of meridional modes the system of equations splits into two systems of two equations. A general iterative technique, asymptotic partitioning of systems of equations, for solving systems of differential equations is presented. As a simple example, Bessel's differential equation is written in matrix form and is solved using this asymptotic technique. Low order solutions for particular examples of a biaxial and uniaxial graded-index fiber are presented. Finally numerical results

obtained using the asymptotic technique are presented for particular examples of isotropic and uniaxial step-index fibers and isotropic, uniaxial and biaxial graded-index fibers.

For purposes of comparison and verification a purely numeric solution method is also presented. The algorithm used by Yeh and Lindgren [17] is improved to handle the case of a uniaxial graded-index fiber.

2. Analytic Solutions

2.1 Introduction

Consider a circularly symmetric optical fiber with the geometry shown in figure 1. The region $0 \leq \rho \leq a$ is referred to as the core and the region $a \leq \rho \leq b$ as the cladding. The permeability of both the core and cladding is μ_0, the permeability of free space. The permittivity of the cladding is $\epsilon_0 \epsilon_c$ where ϵ_0 is the permittivity of free space and ϵ_c is the relative permittivity of the cladding and is assumed to be a constant. The permittivity of core is $\epsilon_0 \bar{\epsilon}_r$ where $\bar{\epsilon}_r$ is the relative permittivity tensor of the core and in general is a function of position in the core. Also it is assumed that the radius of the cladding, b, is sufficiently large so that the fields in the cladding decay exponentially and are essentially equal to zero at $\rho = b$. This eliminates the need to impose boundary conditions at the air-cladding boundary.

Consider the case where the permittivity in the core, $\bar{\epsilon}$ is given by

$$\bar{\epsilon}(\rho) = \epsilon_0 \bar{\epsilon}_r(\rho) = \epsilon_0 \begin{pmatrix} \epsilon_1(\rho) & 0 & 0 \\ 0 & \epsilon_2(\rho) & 0 \\ 0 & 0 & \epsilon_3(\rho) \end{pmatrix}_{\rho,\phi,z} ; \qquad (2-1)$$

where $\epsilon_1(\rho)$, $\epsilon_2(\rho)$ and $\epsilon_3(\rho)$ are the relative permittivities in the ρ, ϕ and z directions respectively.

For time harmonic fields in a source free region, Maxwell's equations can be written as

$$\nabla \times \mathbf{H} = j\omega\epsilon_0\epsilon_r\mathbf{E}, \qquad (2-2)$$

$$\nabla \times \mathbf{E} = -j\omega\mu_0\mathbf{H}, \qquad (2-3)$$

$$\nabla \cdot \mathbf{D} = 0, \qquad (2-4)$$

$$\nabla \cdot \mathbf{B} = 0, \qquad (2-5)$$

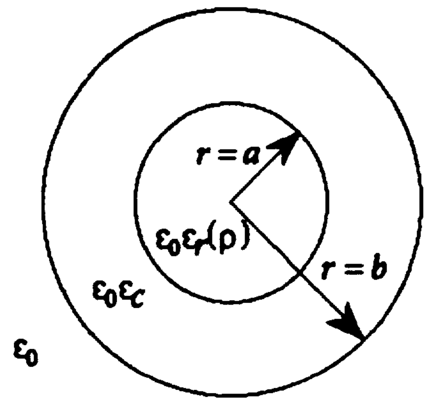

Figure 1 Geometry of the fiber

where ϵ_0 and μ_0 are the permittivity and permeability of free space, and ω is the angular frequency. The problem is to find a solution for eqs. (2-2) to (2-5) in cylindrical coordinates.

If the z and ϕ dependence of the fields is given by

$$e^{-j\beta z+jm\phi},$$

where β is the longitudinal wavenumber and m is any integer (because the fields are periodic in ϕ with period 2π), then eqs. (2-2) and (2-3) can be written in cylindrical coordinates as

$$\frac{m}{\rho}H_z + \beta H_\phi = \omega\epsilon_0\epsilon_1 E_\rho, \qquad (2-2a)$$

$$-j\beta H_\rho - \frac{dH_z}{d\rho} = j\omega\epsilon_0\epsilon_2 E_\phi, \qquad (2-2b)$$

$$\frac{1}{\rho}\frac{d}{d\rho}(\rho H_\phi) - \frac{jm}{\rho}H_\rho = j\omega\epsilon_0\epsilon_3 E_z, \qquad (2-2c)$$

$$\frac{m}{\rho}E_z + \beta E_\phi = -\omega\mu_0 H_\rho, \qquad (2-3a)$$

$$j\beta E_\rho + \frac{dE_z}{d\rho} = j\omega\mu_0 H_\phi, \qquad (2-3b)$$

$$\frac{1}{\rho}\frac{d}{d\rho}(\rho E_\phi) - \frac{jm}{\rho}E_\rho = -j\omega\mu_0 H_z. \qquad (2-3c)$$

2.2 Wave Equation Formulation

2.2.1 Derivation of Differential Equations

Setting $h = Z_0 H$, where $Z_0 = \sqrt{\mu_0/\epsilon_0}$ is the impedance of free space, eqs. (2-2a) and (2-3b) can be written as the following system of equations in the unknowns E_ρ and h_ϕ:

$$k_0\epsilon_1 E_\rho - \beta h_\phi = \frac{m}{\rho} h_z,$$
$$\beta E_\rho - k_0 h_\phi = j\frac{dE_z}{d\rho}, \qquad (2-6)$$

where $k_0 = \omega\sqrt{\epsilon_0\mu_0}$ is the wavenumber of free space. Similarly, eqs. (2-2b) and (2-3a) can be written as:

$$k_0\epsilon_2 E_\phi + \beta h_\rho = j\frac{dh_z}{d\rho},$$
$$\beta E_\phi + k_0 h_\rho = -\frac{m}{\rho} E_z. \qquad (2-7)$$

Solving eqs. (2-6) and (2-7) for E_ρ, h_ϕ, E_ϕ, and h_ρ gives

$$E_\rho = \frac{1}{k_{t1}^2}\left[\frac{mk_0}{\rho} h_z - j\beta\frac{dE_z}{d\rho}\right], \qquad (2-8a)$$

$$h_\phi = \frac{1}{k_{t1}^2}\left[\frac{m\beta}{\rho} h_z - jk_0\epsilon_1\frac{dE_z}{d\rho}\right], \qquad (2-8b)$$

$$E_\phi = \frac{1}{k_{t2}^2}\left[\frac{m\beta}{\rho} E_z + jk_0\frac{dh_z}{d\rho}\right], \qquad (2-8c)$$

$$h_\rho = \frac{1}{k_{t2}^2}\left[\frac{-mk_0\epsilon_2}{\rho} E_z - j\beta\frac{dh_z}{d\rho}\right] \qquad (2-8d)$$

where

$$k_{tn}^2 = k_0^2\epsilon_n - \beta^2, \quad n = 1,2 \qquad (2-9)$$

is the transverse wave number. Eq. (2-8) gives expressions for the transverse field components in terms of the longitudinal components E_z and h_z.

The remaining two equations (2-2c) and (2-3c) can be written as

$$\frac{dE_\phi}{d\rho} + \frac{E_\phi}{\rho} - \frac{jm}{\rho} E_\rho + jk_0 h_z = 0, \qquad (2-10a)$$

$$\frac{dh_\phi}{d\rho} + \frac{h_\phi}{\rho} - \frac{jm}{\rho} h_\rho - jk_0\epsilon_3 E_z = 0. \qquad (2-10b)$$

Substitution of eqs. (2-8a,c) into (2-10a), and eqs. (2-8b,d) into (2-10b) yields:

$$
\begin{aligned}
m\beta\left(\frac{E_z}{k_{t2}^2\rho}\right)' + jk_0\left(\frac{h_z'}{k_{t2}^2}\right)' + \frac{m\beta}{k_{t2}^2\rho^2}E_z + \frac{jk_0}{k_{t2}^2\rho}h_z' \\
- \frac{jm^2k_0}{k_{t1}^2\rho^2}h_z - \frac{m\beta}{k_{t1}^2\rho}E_z' + jk_0h_z = 0, \qquad (2-11a)
\end{aligned}
$$

$$
\begin{aligned}
m\beta\left(\frac{h_z}{k_{t1}^2\rho}\right)' - jk_0\left(\frac{\epsilon_1 E_z'}{k_{t1}^2}\right)' + \frac{m\beta}{k_{t1}^2\rho^2}h_z - \frac{jk_0\epsilon_1}{k_{t1}^2\rho}E_z' \\
+ \frac{jm^2k_0\epsilon_2}{k_{t2}^2\rho^2}E_z - \frac{m\beta}{k_{t2}^2\rho}h_z' - jk_0\epsilon_3E_z = 0 \qquad (2-11b)
\end{aligned}
$$

where $' = d/d\rho$. Simplifying eq. (2-11) by collecting common derivatives of E_z and h_z gives the following

$$
\begin{aligned}
h_z'' + \left(\frac{1}{\rho} - 2\frac{k_{t2}'}{k_{t2}}\right)h_z' + k_{t2}^2\left(1 - \frac{m^2}{k_{t1}^2\rho^2}\right)h_z \\
- \frac{jm\beta}{k_0\rho}\left[\left(1 - \frac{k_{t2}^2}{k_{t1}^2}\right)E_z' - 2\frac{k_{t2}'}{k_{t2}}E_z\right] = 0, \qquad (2-12a)
\end{aligned}
$$

$$
\begin{aligned}
E_z'' + \left[\frac{1}{\rho} + \left(\ln\frac{\epsilon_1}{k_{t1}^2}\right)'\right]E_z' + \frac{\epsilon_3}{\epsilon_1}k_{t1}^2\left(1 - \frac{\epsilon_2}{\epsilon_3}\frac{m^2}{k_{t2}^2\rho^2}\right)E_z \\
+ \frac{jm\beta}{\epsilon_1 k_0\rho}\left[\left(1 - \frac{k_{t1}^2}{k_{t2}^2}\right)h_z' - 2\frac{k_{t1}'}{k_{t1}}h_z\right] = 0. \qquad (2-12b)
\end{aligned}
$$

In general, eqs. (2-12a) and (2-12b) are coupled except for the case $m = 0$. This implies that the general solutions of eqs. (2-12) are of a hybrid type with both $E_z \not\equiv 0$ and $h_z \not\equiv 0$.

Eqs.(2-12) can be written in a more covenient form if we make the following substitutions

$$\frac{\beta}{k_0} = \kappa, \qquad (2-13)$$

$$1 - \frac{k_{t1}^2}{k_{t2}^2} = \frac{\epsilon_2 - \epsilon_1}{\epsilon_2 - \kappa^2}, \quad 1 - \frac{k_{t2}^2}{k_{t1}^2} = \frac{\epsilon_1 - \epsilon_2}{\epsilon_1 - \kappa^2}, \qquad (2-14)$$

$$2\frac{k_{tl}'}{k_{tl}} = \frac{\epsilon_l'}{\epsilon_l - \kappa^2}, \quad l = 1,2 \qquad (2-15)$$

and

$$\left(\ln\frac{\epsilon_1}{k_{t1}^2}\right)' = -\frac{\kappa^2\epsilon_1'}{\epsilon_1(\epsilon_1 - \kappa^2)}. \qquad (2-16)$$

It is also convenient to make a change of variable from ρ to the normalized radius r where $r = \rho/a$ and a is the core radius. Using eqs. (2-13), (2-24) and (2-15), eqs.(2-12) can be rewritten as

$$E_z'' + f_1(r)E_z' + \Lambda^2 g_1(r)E_z = p_2(r)h_z' + q_2(r)h_z, \qquad (2-17a)$$

$$h_z'' + f_2(r)h_z' + \Lambda^2 g_2(r)h_z = p_1(r)E_z' + q_1(r)E_z, \qquad (2-17b)$$

where $' = d/dr$, $\Lambda^2 = (k_0 a)^2$ and

$$f_1(r) = \frac{1}{r} - \frac{\kappa^2 \epsilon_1'(r)}{\epsilon_1(r)[\epsilon_1(r) - \kappa^2]}, \qquad (2-18a)$$

$$f_2(r) = \frac{1}{r} - \frac{\epsilon_2'(r)}{\epsilon_2(r) - \kappa^2}, \qquad (2-18b)$$

$$g_1(r) = \frac{\epsilon_3(r)}{\epsilon_1(r)}[\epsilon_1(r) - \kappa^2]\left[1 - \frac{m^2 \epsilon_2(r)}{\Lambda^2 \epsilon_3(r)[\epsilon_2(r) - \kappa^2]r^2}\right], \qquad (2-18c)$$

$$g_2(r) = [\epsilon_2(r) - \kappa^2]\left[1 - \frac{m^2}{\Lambda^2[\epsilon_1(r) - \kappa^2]r^2}\right], \qquad (2-18d)$$

$$p_1(r) = \frac{jm\kappa}{r}\left[\frac{\epsilon_1(r) - \epsilon_2(r)}{\epsilon_1(r) - \kappa^2}\right], \qquad (2-18e)$$

$$p_2(r) = -\frac{jm\kappa}{\epsilon_1(r)r}\left[\frac{\epsilon_2(r) - \epsilon_1(r)}{\epsilon_2(r) - \kappa^2}\right], \qquad (2-18f)$$

$$q_1(r) = -\frac{jm\kappa}{r}\left[\frac{\epsilon_2'(r)}{\epsilon_2(r) - \kappa^2}\right], \qquad (2-18g)$$

$$q_2(r) = \frac{jm\kappa}{\epsilon_1(r)r}\left[\frac{\epsilon_1'(r)}{\epsilon_1(r) - \kappa^2}\right]. \qquad (2-18h)$$

From eqs. (2-18e) through (2-18h) we can see that the differential equations become decoupled for three particular cases. For so called meridional modes m is equal to zero and from eqs. (2-18e) through (2-18h) it can be seen that p_1, p_2, q_1 and q_2 are also zero. For an isotropic and uniaxial step index fibers ϵ_1 and ϵ_2 are equal and constant, therefore, from eqs. (2-18e) through (2-18h) p_1, p_2, q_1 and p_2 are identically equal to zero.

2.2.2 Derivation of Dispersion Relation

For the region $r < 1$, let the general solutions of eqs. (2-17a) and (2-17b) be given by

$$E_z = Ae(r), \quad h_z = Bh(r), \tag{2-19}$$

where A and B are constants. Using eqs. (2-8b,c) in eqs. (2-8b) and (2-8c) the tangential components rE_ϕ and rh_ϕ can be written as

$$rE_\phi = \frac{m\beta}{ak_{t2}^2}Ae(r) + \frac{jk_0r}{ak_{t2}^2}Bh'(r) \tag{2-20a}$$

$$rh_\phi = \frac{m\beta}{ak_{t1}^2}Bh(r) - \frac{jk_0\epsilon_1 r}{ak_{t1}^2}Ae'(r) \tag{2-20b}$$

where $e'(r) = (d/dr)e(r)$ and $h'(r) = (d/dr)h(r)$.

For the region $r > 1$, ϵ_1, ϵ_2, and ϵ_3 are equal to a constant ϵ_c. Under these conditions eqs. (2-17a) and (2-17b) simplify to Bessel's equations of the variable $k_t ar$ where $k_t^2 = k_0^2\epsilon_c - \beta^2$. For guided modes we require that $\beta^2 \geq k_0^2\epsilon_c$ and that the field be of the form $e^{-\gamma r}$ as r tends to infinity, with $\gamma > 0$. If we let $\gamma^2 = -k_t^2$ we can choose $K_m(\gamma ar)$, the modified Bessel function of the second kind, as the solution which satisfies the requirement of a decaying exponential. E_z and h_z can then be given by

$$E_z = CK_m(\gamma ar), \quad h_z = DK_m(\gamma ar), \tag{2-21}$$

where C and D are constants and $\gamma^2 = -k_t^2 = \beta^2 - k_0^2 e_c$. From eqs. (2-8b) and (2-8c) the tangential components rE_ϕ and rh_ϕ for $r > 1$ are given by

$$rE_\phi = -\frac{m\beta}{a\gamma^2}CK_m(\gamma ar) - \frac{jk_0r}{a\gamma}DK'_m(\gamma ar) \tag{2-22a}$$

$$rh_\phi = -\frac{m\beta}{a\gamma^2}DK_m(\gamma ar) + \frac{jk_0\epsilon_c r}{a\gamma}CK'_m(\gamma ar) \tag{2-22b}$$

where $K'_m(\gamma ar) = dK_m(\gamma ar)/d(\gamma ar)$.

At $r = 1$ the tangential components of the electric and magnetic fields, E_z, h_z, E_ϕ and h_ϕ must be continuous. Using eqs. (2-19) to (2-22) the boundary condition can be written as

$$\begin{pmatrix} e(1) & 0 & -K_m(\gamma a) & 0 \\ 0 & h(1) & 0 & -K_m(\gamma a) \\ \frac{m\beta}{ak_{t2}^2(1)}e(1) & \frac{jk_0}{ak_{t2}^2(1)}h'(1) & \frac{m\beta}{a\gamma^2}K_m(\gamma a) & \frac{jk_0}{\gamma}K'_m(\gamma a) \\ -\frac{jk_0\epsilon_1}{ak_{t1}^2(1)}e'(1) & \frac{m\beta}{ak_{t1}^2(1)}h(1) & -\frac{jk_0\epsilon_c}{\gamma}K'_m(\gamma a) & \frac{m\beta}{a\gamma^2}K_m(\gamma a) \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad (2-23)$$

For a non-trivial solution to eq.(2-23) the determinant

$$\Delta = \begin{vmatrix} e(1) & 0 & -K_m(\gamma a) & 0 \\ 0 & h(1) & 0 & -K_m(\gamma a) \\ \frac{m\beta}{ak_{t2}^2(1)}e(1) & \frac{jk_0}{ak_{t2}^2(1)}h'(1) & \frac{m\beta}{a\gamma^2}K_m(\gamma a) & \frac{jk_0}{\gamma}K'_m(\gamma a) \\ -\frac{jk_0\epsilon_1}{ak_{t1}^2(1)}e'(1) & \frac{m\beta}{ak_{t1}^2(1)}h(1) & -\frac{jk_0\epsilon_c}{\gamma}K'_m(\gamma a) & \frac{m\beta}{a\gamma^2}K_m(\gamma a) \end{vmatrix} \qquad (2-24)$$

must be identically equal to zero.

For convenience let $\epsilon = e(1)$, $h = h(1)$, $e' = e'(1)$, $h' = h'(1)$, $k_{tn}^2(1) = k_{tn}^2$, $K_m = K_m(\gamma a)$, and $K'_m = K'_m(\gamma a)$. By expanding the determinant and performing some algebraic manipulations the generalized dispersion relation is given by

$$\left(\frac{m\beta}{k_0}\right)^2 \left[\frac{1}{(\gamma a)^2} + \frac{1}{(k_{t1}a)^2}\right] \left[\frac{1}{(\gamma a)^2} + \frac{1}{(k_{t2}a)^2}\right] = \left[\frac{\epsilon_c}{\gamma a}\frac{K'_m}{K_m} + \frac{\epsilon_1}{(k_{t1}a)^2}\frac{e'}{e}\right] \left[\frac{1}{\gamma a}\frac{K'_m}{K_m} + \frac{1}{(k_{t2}a)^2}\frac{h'}{h}\right] \qquad (2-25)$$

2.2.3 Exact Solutions

For an isotropic step index fiber ϵ_1, ϵ_2 and ϵ_3 are all equal to the constant ϵ_r. Eqs. (2-17) then simplify to

$$E_z'' + \frac{1}{r}E_z' + \left[(k_t a)^2 - \frac{m^2}{r^2}\right]E_z = 0, \qquad (2-26a)$$

$$h_z'' + \frac{1}{r}h_z' + \left[(k_t a)^2 - \frac{m^2}{r^2}\right]h_z = 0, \qquad (2-26b)$$

where $(k_t a)^2 = \Lambda^2(\epsilon_r - \kappa^2)$ or $k_t^2 = \epsilon_r k_0^2 - \beta^2$. E_z and h_z are then given by

$$E_z = AJ_m(k_t ar)\,, \quad h_z = BJ_m(k_t ar), \qquad (2-27)$$

where A and B are constants. By substituting eq. (2-27) into the generalized dispersion relation given by eq. (2-25) and making use of the fact that for a step index fiber $k_{t1}^2 = k_{t2}^2 = k_t^2$ gives the well known dispersion relation for a step index fiber:

$$\left(\frac{m\beta}{k_0}\right)^2\left[\frac{1}{(a\gamma)^2} + \frac{1}{(ak_t)^2}\right]^2 = \left[\frac{\epsilon_c}{a\gamma}\frac{K_m'(\gamma a)}{K_m(\gamma a)} + \frac{\epsilon_r}{ak_t}\frac{J_m'(k_t a)}{J_m(k_t a)}\right] \cdot \left[\frac{1}{a\gamma}\frac{K_m'(\gamma a)}{K_m(\gamma a)} + \frac{1}{ak_t}\frac{J_m'(k_t a)}{J_m(k_t a)}\right], \qquad (2-28)$$

where $\gamma^2 = \beta^2 - \epsilon_c k_0^2$.

For a uniaxial step index fiber $\epsilon_1 = \epsilon_2 \neq \epsilon_3$ and ϵ_1 and ϵ_3 are constants. Eqs. (2-17) simplify to

$$E_z'' + \frac{1}{r}E_z' + \left[\frac{\epsilon_3}{\epsilon_1}(k_t a)^2 - \frac{m^2}{r^2}\right]E_z = 0 \qquad (2-29a)$$

$$h_z'' + \frac{1}{r}h_z' + \left[(k_t a)^2 - \frac{m^2}{r^2}\right]h_z = 0 \qquad (2-29b)$$

where $(k_t a)^2 = \Lambda^2(\epsilon_1 - \kappa^2)$ or $k_t^2 = \epsilon_1 k_0^2 - \beta^2$. By defining an anisotropy parameter $p^2 = \epsilon_3/\epsilon_1$, the solutions of eqs. (2-29a) and (2-29b) are given by

$$E_z = AJ_m(pk_t ar)\,, \quad h_z = BJ_m(k_t ar), \qquad (2-30)$$

where A and B are constants. By using eq. (2-30) in eq. (2-25) and making use of the fact that $k_{t1}^2 = k_{t2}^2 = k_t^2$ the dispersion relation for a uniaxial step index fiber is given by [2]

$$\left(\frac{m\beta}{k_0}\right)^2 \left[\frac{1}{(a\gamma)^2} + \frac{1}{(ak_t)^2}\right]^2 = \left[\frac{\epsilon_c}{a\gamma}\frac{K_m'(\gamma a)}{K_m(\gamma a)} + \frac{\sqrt{\epsilon_1\epsilon_3}}{ak_t}\frac{J_m'(pk_t a)}{J_m(pk_t a)}\right] \cdot \left[\frac{1}{a\gamma}\frac{K_m'(\gamma a)}{K_m(\gamma a)} + \frac{1}{ak_t}\frac{J_m'(k_t a)}{J_m(k_t a)}\right] \qquad (2-31)$$

A representative case for both an isotropic and a uniaxial step-index fiber is presented. When $m = 0$ the solutions of the dispersion relations, either eq. (2-29) or (2-31), are either transverse electric, $E_z = 0$ or transverse magnetic, $h_z = 0$ and are designated by the notation TE_{0n} and TM_{0n} respectively where $n = 1, 2, 3, \ldots$. When $m > 0$ the electric and magnetic fields for all solutions have components in the axial direction, i.e $E_z \not\equiv 0$ and $h_z \not\equiv 0$ and are therefore designated as hybrid modes. A hybrid mode is arbitrarily designated as EH (HE) if at some arbitrary reference point E_z (h_z) makes a larger contribution than h_z (E_z) to the transverse field. A less arbitrary classification scheme, which gives the same mode designations, based on the ratio of H_z to E_z at cutoff has been proposed by Snitzer[1] and refined by Safaai and Yip[19].

As an example of an isotropic step-index fiber the relative permittivities of the core and cladding are taken to be $\epsilon_r = n_r^2$ and $\epsilon_c = n_c^2$ respectively where $n_r = 1.515$ is the refractive index of the core and $n_c = 1.5$ is the refractive index of the cladding. Figures 2 and 3 are plots of the normalized propagation constant, $\kappa = \beta/k_0$, versus the normalized free space wavenumber, $\Lambda = k_0 a$ for the cases $m = 0$ and $m = 1$ respectively. Two notable features are that the TE_{0n} and the TM_{0n} modes are essentially degenerate except close to cutoff and all modes except the HE_{11} mode have a finite non-zero cutoff frequency.

As an example of a uniaxial step-index fiber the relative permittivities in the core and

cladding are taken to be $\epsilon_1 = \epsilon_2 = n_1^2$, $\epsilon_3 = n_3^2$ and $\epsilon_c = n_c^2$ where $n_1 = 1.515$ is the refractive index of the core in the ρ and ϕ directions, $n_3 = 2$ is the refractive index of the core in the z direction and $n_c = 1.5$. Figures 4 and 5 are plots of κ versus $k_0 a$ for the cases $m = 0$ and $m = 1$. A comparison of eqs. (2-27) and (2-30) implies that the introduction of anisotropy into a step index fiber affects modes where E_z makes the larger contribution to the transverse fields, i.e. TM_{0n} and EH_{mn} modes. A comparison of figures 2 and 4 show that the TE_{0n} modes for the isotropic and uniaxial step-index fibers are identical while the TM_{0m} modes for the uniaxial case are displaced from the corresponding TM_{0m} for the isotropic case. Comparing figures 3 and 5 it can be seen that both the EH and HE modes for the uniaxial fiber are displaced from the corresponding mode for the isotropic fiber. As expected the effect of the anisotropy is much more pronounced in the EH modes than in the HE modes.

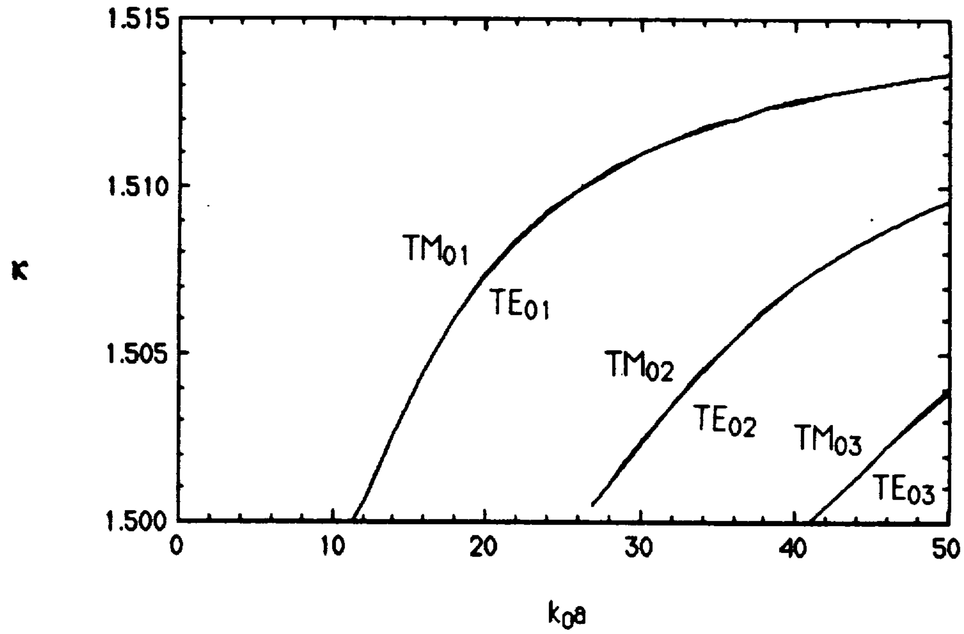

Figure 2 Isotropic step-index fiber: $m = 0$

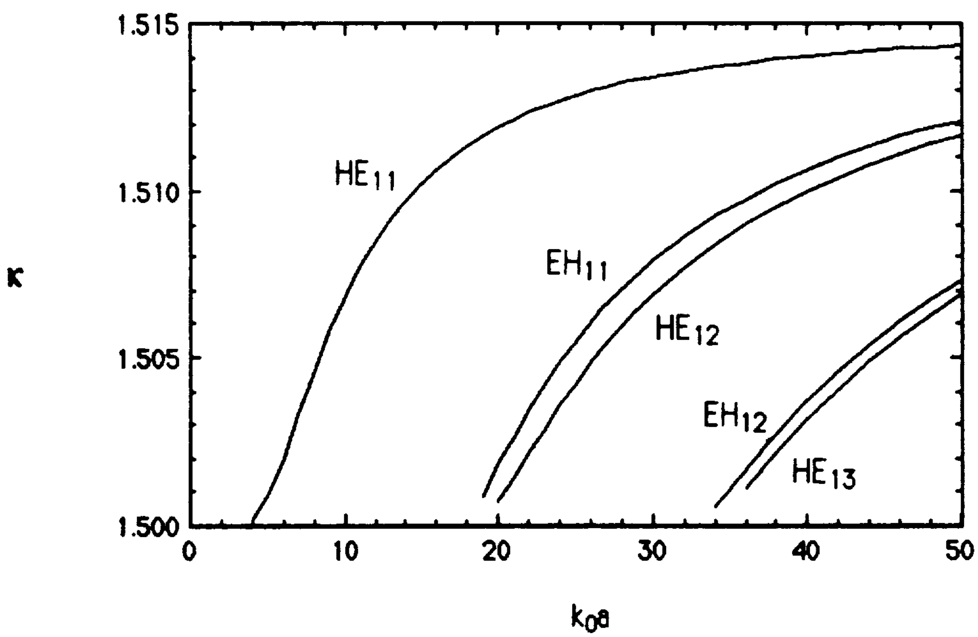

Figure 3 Isotropic step-index fiber: $m = 1$

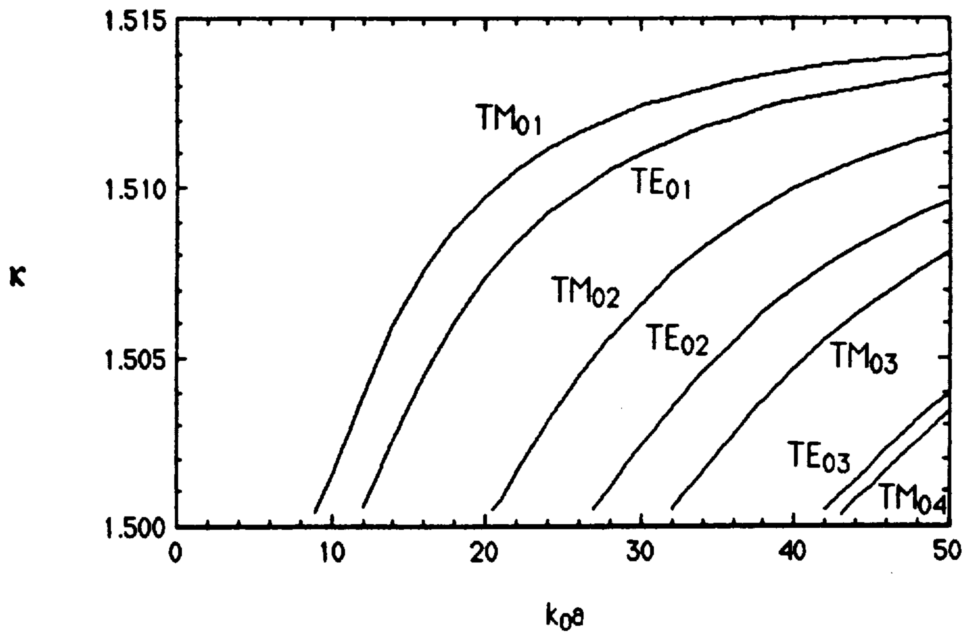

Figure 4 Uniaxial step-index fiber: $m = 0$

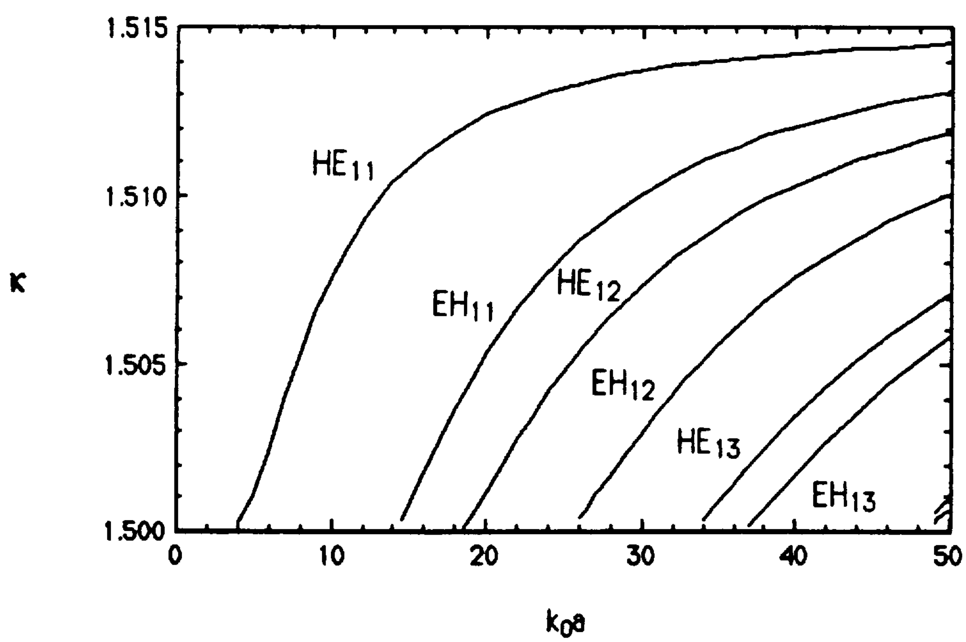

Figure 5 Uniaxial step-index fiber: $m = 1$

2.2.4 WKB Solutions

In order to solve eq. (2-17) for the case of an inhomogeneous fiber some simplifications must be made. If we assume the variation in the permittivities is very small over distances of one wavelength and the core is infinite in extent (eliminates need to impose boundary conditions) then the WKB method can be used. For the case of an isotropic or uniaxial graded-index fiber $\epsilon_1(r) = \epsilon_2(r)$ and eq. (2-17) simplifies into

$$E_z'' + \frac{1}{r}E_z' + \Lambda^2 g_1(r) E_z = 0, \qquad (2-32a)$$

$$h_z'' + \frac{1}{r}h_z' + \Lambda^2 g_2(r) h_z = 0, \qquad (2-32b)$$

where g_1 and g_2 are given by

$$g_1(r) = \frac{\epsilon_3(r)}{\epsilon_1(r)}\left[\epsilon_1(r) - \kappa^2\right] - \frac{m^2}{\Lambda^2 r^2}, \qquad (2-33a)$$

$$g_2(r) = \epsilon_1(r) - \kappa^2 - \frac{m^2}{\Lambda^2 r^2}. \qquad (2-33b)$$

Let E_z be of the form

$$E_z = e^{jk_0\psi(r)} \qquad (2-34a)$$

then

$$E_z' = jk_0\psi' E_z, \qquad (2-34b)$$

$$E_z'' = \left[jk_0\psi'' - k_0^2(\psi')^2\right]E_z, \qquad (2-34c)$$

Substituting eq. (2-34) into (2-32a) and dropping common factor of E_z gives the following differential equation for $\psi(r)$

$$jk_0\psi'' - k_0^2(\psi')^2 + \frac{jk_0}{r}\psi' + \Lambda^2 g_1 = 0. \qquad (2-35)$$

If g_1 is a slowly varying functon of r then $\psi(r)$ can be approximated by

$$\psi(r) \approx \psi_0(r) + \frac{1}{k_0}\psi_1(r); \tag{2 – 36a}$$

and

$$\psi' = \psi_0' + \frac{1}{k_0}\psi_1', \tag{2 – 36b}$$

$$(\psi')^2 = (\psi_0')^2 + \frac{2}{k_0}\psi_0'\psi_1' + \frac{1}{k_0^2}(\psi_1')^2, \tag{2 – 36c}$$

$$\psi'' = \psi_0'' + \frac{1}{k_0}\psi_1''. \tag{2 – 36d}$$

Using eqs. (2-36a) through (2-36d) in eq. (2-35) gives the following equation relating ψ_0 and ψ_1 to the functions f_1 and g_1.

$$\begin{aligned} jk_0\psi_0'' + j\psi_1'' - k_0^2(\psi_0')^2 - 2k_0\psi_0'\psi_1' - (\psi_1')^2 & \\ + \frac{jk_0}{r}\psi_0' + \frac{j}{r}\psi_1' + \Lambda^2 g_1 = 0 & \end{aligned} \tag{2 – 37}$$

Recalling that $\Lambda^2 = (k_0 a)^2$, if we equate like powers of k_0 we obtain the following equations for ψ_0 and ψ_1:

$$(\psi_0')^2 - a^2 g_1 = 0, \tag{2 – 38a}$$

$$j\psi_0'' - 2\psi_0'\psi_1' + \frac{j}{r}\psi_0' = 0. \tag{2 – 38b}$$

Solving eq. (2-38a) gives

$$\psi_0 = \pm a \int \sqrt{g_1(r)}\, dr \tag{2 – 39}$$

Eq. (2-38b) can be written as

$$j\frac{\psi_0''}{\psi_0'} - 2\psi_1' + \frac{j}{r} = 0,$$

or

$$\psi_1(r) = \frac{j}{2}\ln(r\psi_0'). \tag{2 – 40}$$

Using eqs. (2-39) and (2-40) E_z can be written as

$$E_z = \frac{e^{\pm jk_0 a \int \sqrt{g_1(r)}\,dr}}{\sqrt{r}[a^2 g_1(r)]^{1/4}}. \qquad (2-41)$$

Using the same method h_z can be found to be

$$h_z = \frac{e^{\pm jk_0 a \int \sqrt{g_2(r)}\,dr}}{\sqrt{r}[a^2 g_2(r)]^{1/4}} \qquad (2-42)$$

The mode condition for a WKB solution [20],[6] requires that

$$k_0 a \int_{r_1}^{r_2} \sqrt{g_i(r)}\,dr = (n + \frac{1}{2})\pi \quad n = 0, 1, 2, \ldots \qquad (2-43)$$

where $i = 1, 2$ and r_1 and r_2 are the turning points (zeroes) of g_i. An exact solution of eq. (2-43) is possible only for a small number of permittivity profiles. In general, eq. (2-43) must be solved numerically to determine the allowable modes.

Consider the case where the permittivity profiles in the core are given by

$$\epsilon_i(r) = \epsilon_i\Big[1 - 2\Delta_i r^{\alpha_i}\Big] \qquad i = 1, 2, 3 \qquad (2-44)$$

where α_i is a parameter which describes the shape of the permittivity profile,

$$\Delta_i = \frac{\epsilon_i - \epsilon_r}{2\epsilon_i} \qquad i = 1, 2, 3 \qquad (2-43)$$

and ϵ_i is the relative permittivity at the center of the core. The value of the parameter α_i must be greater than or equal to one. Note that in the limit $\alpha_i \to \infty$ the permittivity profile approaches the profile for a step index fiber.

Let us consider the special case of an isotropic graded-index fiber.with a parabolic profile. Since $\epsilon_1(r) = \epsilon_2(r) = \epsilon_3(r) = \epsilon_r(r)$ eq. (2-33) reduces to

$$g_1(r) = g_2(r) = g(r) = \epsilon_r(r) - \kappa^2 - \frac{m^2}{\Lambda^2 r^2} \qquad (2-46)$$

where

$$\epsilon_r(r) = \epsilon_r\left[1 - 2\Delta r^2\right] \qquad (2-47)$$

and

$$\Delta = \frac{\epsilon_r - \epsilon_c}{2\epsilon_r} \qquad (2-48)$$

For this choice of $\epsilon_r(r)$ it is possible to analytically solve eq. (2-43) to obtain the allowable modes.

The turning points r_1 and r_2, determined by setting $g(r) = 0$, are given by

$$r_2 = -r_1 = \sqrt{\frac{\epsilon_r - \kappa^2}{2\epsilon_r\Delta}} \qquad m = 0 \qquad (2-49a)$$

$$r_{1,2} = \sqrt{\frac{1}{2}\left[A \pm \sqrt{A^2 - 4B}\right]} \qquad m \neq 0 \qquad (2-49b)$$

where

$$A = \frac{\epsilon_r - \kappa^2}{2\epsilon_r\Delta} \qquad (2-50a)$$

and

$$B = \frac{m^2}{2\epsilon_r\Delta\Lambda^2} \qquad (2-50b)$$

When $m = 0$, substituting eq. (2-46) into eq. (2-43) and integrating, using r_1 and r_2 given by eq. (2-49a), results in the following mode condition

$$\Lambda\sqrt{2\epsilon_r\Delta}\left(\frac{\epsilon_r - \kappa^2}{2\epsilon_r\Delta}\right)\frac{\pi}{2} = (n + \frac{1}{2})\pi \qquad m = 0\,. \qquad (2-51)$$

Solving eq. (2-51) for κ gives

$$\kappa = \frac{\beta}{k_0} = \sqrt{\epsilon_r - \frac{\sqrt{2\epsilon_r\Delta}}{\Lambda}(2n+1)} \qquad m = 0\,. \qquad (2-52)$$

Similarly, when $m \neq 0$ substituting eq. (2-46) into eq. (2-43) and integrating gives

$$\frac{\Lambda\sqrt{2\epsilon_r\Delta}}{2}\Big(\frac{r_1^2+r_2^2}{2}-r_1r_2\Big)\pi = (n+\frac{1}{2})\pi \qquad m \neq 0. \qquad (2-53)$$

Substituting for r_1 and r_2 from eq. (2-49b) and solving for κ results in

$$\kappa = \frac{\beta}{k_0} = \sqrt{\epsilon_r - \frac{2\sqrt{2\epsilon_r\Delta}}{\Lambda}(|m|+2n+1)} \qquad m \neq 0\,. \qquad (2-54)$$

Plots of κ versus k_0a for the case $m = 0, 1$ and 2, when $n_r = 1.515$ and $n_c = 1.5$ are shown in figures 5, 6 and 7. At the present time these WKB solutions will be designated by the notation WKB_{mn} where m and n correspond to the m and n in eqs. (2-52) and (2-53).

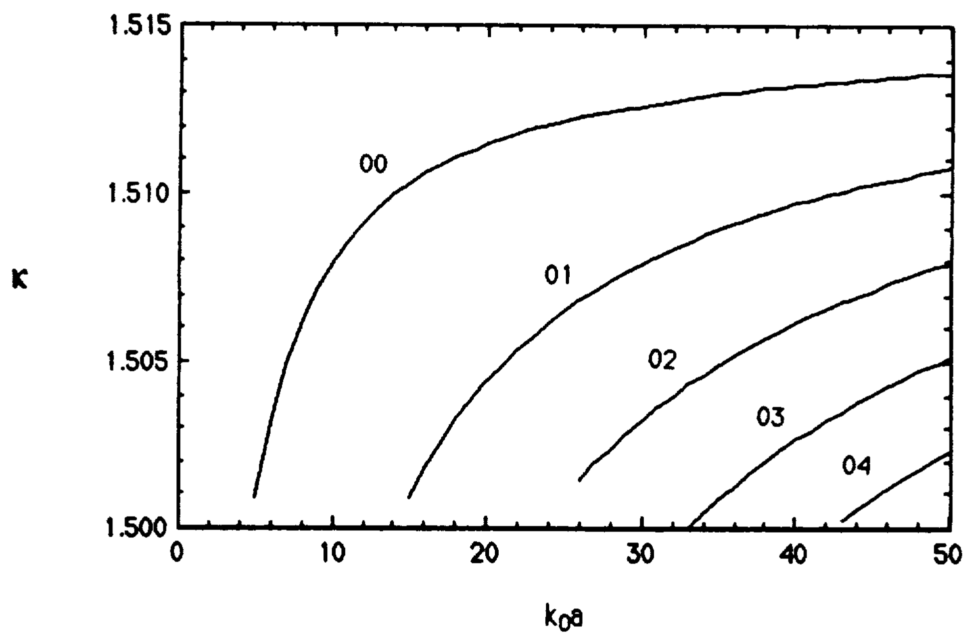

Figure 6 WKB solution for an isotropic graded-index fiber: m=0

For the case of a uniaxial graded-index fiber $\epsilon_1(r) \neq \epsilon_3(r)$ and the functions $g_1(r)$ and $g_2(r)$, given by eq. (2-33) are not equal. Comparing eqs. (2-33b) and (2-46) it is clear that if

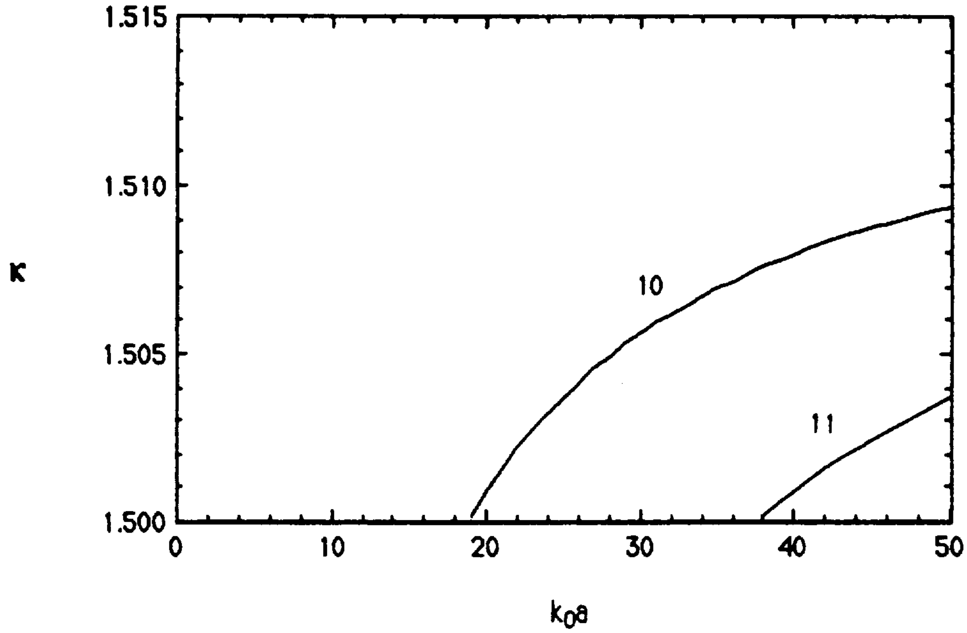

Figure 7 WKB solution for an isotropic graded-index fiber: m=1

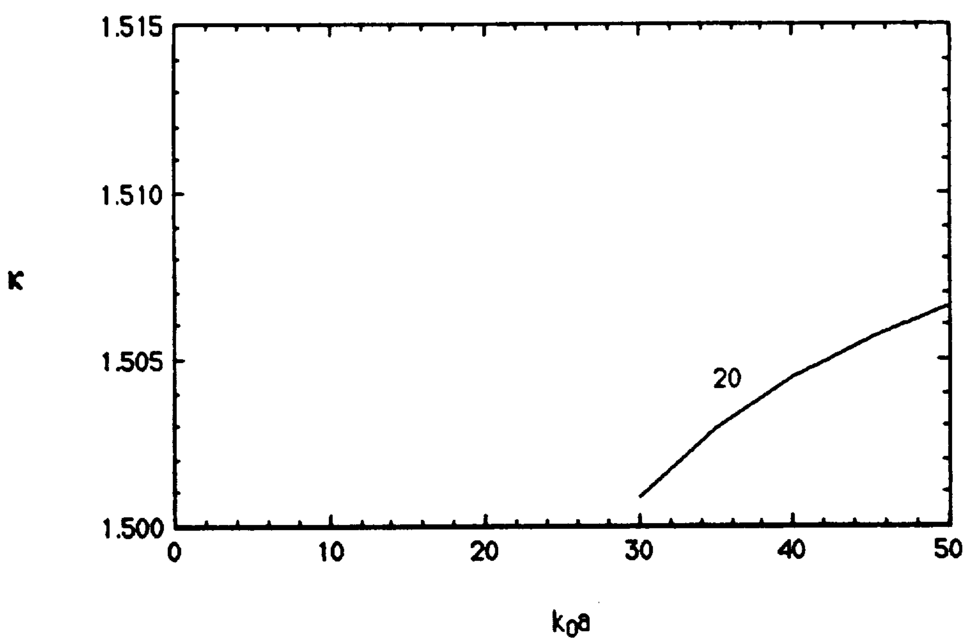

Figure 8 WKB solution for an isotropic graded-index fiber: m=2

$\epsilon_1(r)$ in eq. (2-33b) is equal to $\epsilon(r)$ in eq. (2-46) then the solutions of the modal condition, eq. (2-43), when $i = 2$ are identical to the solutions for the isotropic case.

Again, consider the case where the relative permittivities in the core have parabolic profiles. The solution of the mode condition, eq. (2-46), when $i = 1$ must be found by numerical integration. These solutions corresponding to the solutions of eq. (2-32a) for E_z and will be designated as E_{mn} modes. The solution of the mode condition when $i = 2$ are identical to the solutions for an isotropic graded-index fiber given by eqs. (2-52) and (2-53). These solutions correspond to solutions of eq. (2-32b) for h_z and will be designated as H_{mn} modes. Figures 9 and 10 are plots of κ versus $k_0 a$ for a uniaxial graded-index fiber for the cases $m = 0$ and 1 with $n_1 = 1.515$, $n_3 = 2$ and $n_c = 1.5$.

It is important to remember that the E_z and h_z given by eqs. (2-41) and (2-42) are not solutions of the complete vector problem given by eqs. (2-17) and (2-18) but are rather solutions of a related scalar problem given by eqs. (2-32) and (2-33). Assuming an infinite core, an alternative solution of the scalar problem for an isotropic parabolic-index fiber [4], [21] is given by

$$E_z, h_z = B_{mn}\left(\frac{\rho}{s_0}\right)^m L_n^m\left(\frac{\rho^2}{s_0^2}\right)e^{-\frac{1}{2}\left(\frac{\rho}{s_0}\right)^2} \qquad (2-55)$$

and

$$\beta_{mn}^2 = k_0^2\epsilon_r - \frac{2}{s_0^2}(m + 2n + 1) \qquad (2-56)$$

where $m = 0, 1, 2, \ldots$, $n = 0, 1, 2, \ldots$, s_0 is the characteristic spot size of the medium defined as $s_0^2 = a/k_0\sqrt{2\epsilon_r\Delta}$, L_n^m is a generalized Laguerre polynomial and B_{mn} is a modal constant. It can be readily seen that eqs. (2-54) and (2-56) are identical expressions for the propagation constant β.

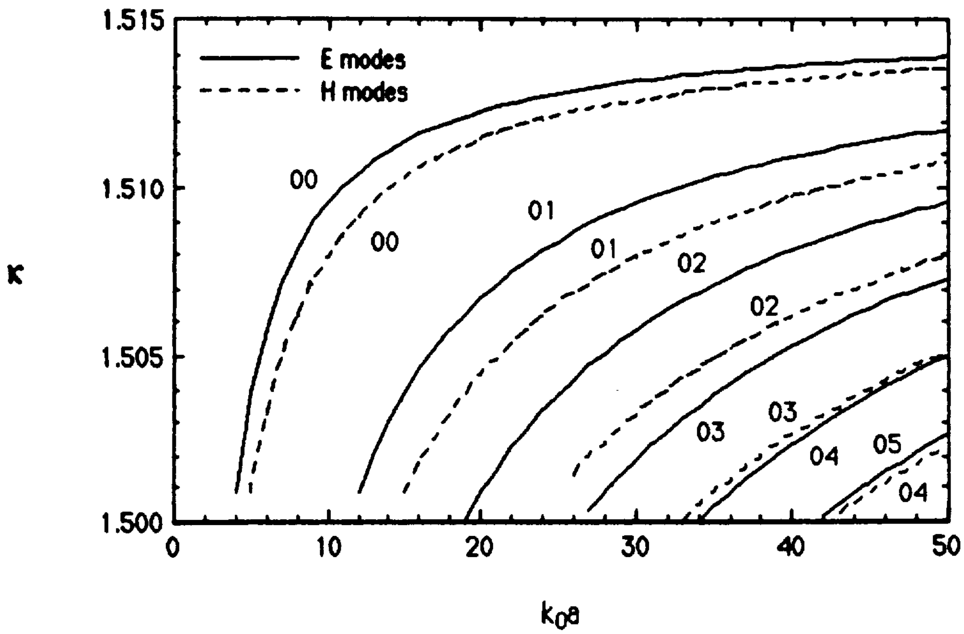

Figure 9 WKB solution for a uniaxial graded-index fiber: m=0

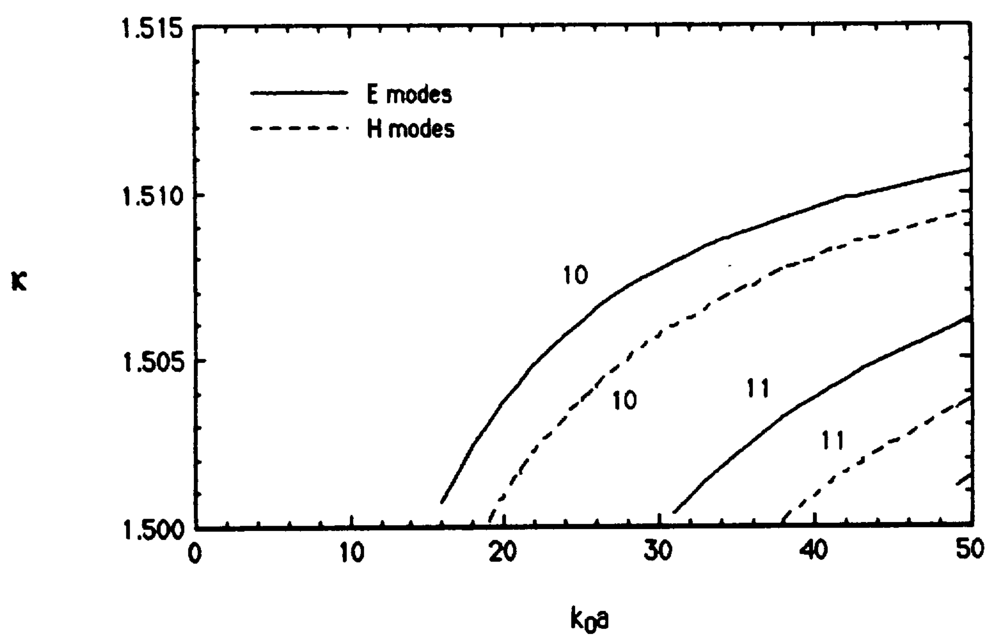

Figure 10 WKB solution for a uniaxial graded-index fiber: m=1

Assuming an infinite core and the fields are far from cutoff a vector analysis of an isotropic parabolic-index fiber [7], [9] gives the following for the transverse fields:

$$E_t^{(i)} = (\mp j\hat{\rho} - \hat{\phi})\Psi^{(i)} \quad \text{and} \quad H_t^{(i)} = \frac{\sqrt{\epsilon_r}}{Z_0}\hat{z} \times E_t^{(i)} \qquad (2-56)$$

where

$$\Psi^{(i)} = c_i \left(\frac{\rho}{s_0}\right)^{m\mp 1} L_{n-1}^{m\mp 1}\left(\frac{\rho^2}{s_0^2}\right) e^{-\frac{1}{2}\left(\frac{\rho}{s_0}\right)^2} \qquad (2-57)$$

and

$$\beta_{mn}^2 = \begin{cases} k_0^2\epsilon_r - \frac{2}{s_0^2}[m + 2(n-1)] & i = 1; \\ k_0^2\epsilon_r - \frac{2}{s_0^2}(m + 2n), & i = 2, \end{cases} \qquad (2-58)$$

where $m = 1, 2, 3, \ldots$, $n = 1, 2, 3, \ldots$, and the upper(lower) sign corresponds to $i = 1 (i = 2)$. When $i = 1$ it can be shown that the solutions correspond to HE modes while when $i = 2$ the solutions correspond to EH modes. For the special case $m = 0$, meridional modes, it can be shown for the TE_{0n} modes that

$$E_\rho = 0 \quad \text{and} \quad E_\phi = -\Psi^{(2)} \qquad (2-59a)$$

while for the TM_{0n}

$$E_\rho = \Psi^{(2)} \quad \text{and} \quad E_\phi = 0 \qquad (2-59b)$$

where for both TE_{0n} and TM_{0n} modes

$$\beta_{0n}^2 = k_0^2\epsilon_r - \frac{4n}{s_0^2}\,. \qquad (2-60)$$

Comparing the scalar solutions given by eqs. (2-55) and (2-56) with the vector solutions given by eqs. (2-57), (2-58), (2-59) and (2-60) it can be seen that

$$\beta_{m,n}^{\text{vector}} = \begin{cases} \beta_{m-1,n-1}^{\text{scalar}} & \text{for HE}_{mn} \text{ modes;} \\ \beta_{m+1,n-1}^{\text{scalar}} & \text{for TE}_{0n}\text{, TM}_{0n} \text{ and EH}_{mn} \text{ modes,} \end{cases} \qquad (2-61)$$

for $m = 0, 1, 2, \ldots$, and $n = 1, 2, 3, \ldots$ where $\beta_{m,n}^{\text{vector}}$ is given by eq. (2-58) and $\beta_{m,n}^{\text{scalar}}$ is given by eq. (2-56).

2.3 Matrix Differential Equation Formulation

2.3.1 Derivation of Differential Equation

In general the solution of eq. (2-17) is not possible except for the case of an isotropic or uniaxial core. A direct series solution for a more general case is not possible except for the case when $m = 0$. However, a series solution in this case still may not be possible due to the poles in $f_1(r)$, $f_2(r)$, $g_1(r)$ and $g_2(r)$. The WKB solution of eq. (2-17) while useful for determining propagation constants away from cutoff is essentially the solution of a scalar wave equation. It also ignores the effects of electromagnetic boundary conditions and the effects of coupling between E_z and h_z.

An alternative formulation [18] is to write eqs. (2-2) and (2-3) as a set of four first order differential equations in terms of the tangential field components. This formulation preserves the vector nature of this problem and permits the use of the boundary conditions.

Eqs. (2-2) and (2-3) can be rewritten as

$$E_\rho = \frac{1}{\omega\epsilon_0\epsilon_1}\left[\frac{m}{\rho}H_z + \beta H_\phi\right], \tag{2 - 62a}$$

$$H_\rho = -\frac{1}{\omega\mu_0}\left[\frac{m}{\rho}E_z + \beta H_\phi\right], \tag{2 - 62b}$$

and

$$\frac{dE_z}{d\rho} = j\omega\mu_0 H_\phi - j\beta E_\rho \tag{2 - 63a}$$

$$\frac{d}{d\rho}(\rho E_\phi) = jmE_\rho - j\omega\mu_0\rho H_z \tag{2 - 63b}$$

$$\frac{dH_z}{d\rho} = -j\omega\epsilon_0\epsilon_2 E_\phi - j\beta H_\rho \tag{2 - 63c}$$

$$\frac{d}{d\rho}(\rho H_\phi) = jmH_\rho + j\omega\epsilon_0\epsilon_3\rho E_z \tag{2 - 63d}$$

where eqs. (2-62a) and (2-62b) represent two scalar equations and eqs. (2-63a,b,c,d) represent four first order differential equations. Substituting eq. (2-62) into eq. (2-63), recognizing that $k_0 = \omega\sqrt{\epsilon_0\mu_0}$, $Z_0 = \sqrt{\epsilon_0/\mu_0}$ and $\kappa = \beta/k_0$ and making a change of variable from ρ to a normalized radius s, where $s = k_0\rho = (k_0a)r = \Lambda r$ gives

$$\frac{dE_z}{ds} = -j\frac{m\kappa}{s\epsilon_1}h_z + \frac{j}{s\epsilon_1}(\epsilon_1 - \kappa^2)(sh_\phi), \qquad (2-64a)$$

$$\frac{d}{ds}(sE_\phi) = \frac{j}{s\epsilon_1}(m^2 - \epsilon_1 s^2)h_z + j\frac{m\kappa}{s\epsilon_1}(sh_\phi), \qquad (2-64b)$$

$$\frac{dh_z}{ds} = j\frac{m\kappa}{s}E_z - \frac{j}{s}(\epsilon_2 - \kappa^2)(sE_\phi), \qquad (2-64c)$$

$$\frac{d}{ds}(sh_\phi) = -\frac{j}{s}(m^2 - \epsilon_3 s^2)E_z - j\frac{m\kappa}{s}(sE_\phi). \qquad (2-64d)$$

Eq. (2-64) can be rewritten in matrix form as

$$\frac{d\mathbf{u}}{ds} = \frac{1}{s}\mathbf{A}(s)\mathbf{u}, \qquad (2-65a)$$

where

$$\mathbf{u} = (\, E_z \quad sE_\phi \quad h_z \quad sh_\phi\,)^T \qquad (2-65b)$$

and

$$\mathbf{A}(s) = \begin{pmatrix} 0 & 0 & -j\frac{m\kappa}{\epsilon_1} & \frac{j}{\epsilon_1}(\epsilon_1 - \kappa^2) \\ 0 & 0 & \frac{j}{\epsilon_1}(m^2 - \epsilon_1 s^2) & j\frac{m\kappa}{\epsilon_1} \\ jm\kappa & -j(\epsilon_2 - \kappa^2) & 0 & 0 \\ -j(m^2 - \epsilon_3 s^2) & -jm\kappa & 0 & 0 \end{pmatrix}. \qquad (2-65c)$$

For the special case of meridional modes, $m = 0$, eqs. (2-64) can be separated into two systems each containing two equations. The first set corresponding to transverse magnetic modes can be written in matrix form as

$$\frac{d\mathbf{u}^{(\mathrm{TM})}}{ds} = \frac{1}{s}\mathbf{A}^{(\mathrm{TM})}(s)\mathbf{u}^{(\mathrm{TM})} \qquad (2-66a)$$

where

$$\mathbf{u}^{(\mathrm{TM})} = (\, E_z \quad sh_\phi \,)^T \tag{2-66b}$$

and

$$\mathbf{A}^{(\mathrm{TM})}(s) = \begin{pmatrix} 0 & \frac{j}{\epsilon_1}(\epsilon_1 - \kappa^2) \\ j\epsilon_3 s^2 & 0 \end{pmatrix} \tag{2-66c}$$

The second set corresponding to transverse electric modes can be written as

$$\frac{d\mathbf{u}^{(\mathrm{TE})}}{ds} = \frac{1}{s}\mathbf{A}^{(\mathrm{TE})}(s)\mathbf{u}^{(\mathrm{TE})} \tag{2-67a}$$

where

$$\mathbf{u}^{(\mathrm{TE})} = (\, h_z \quad sE_\phi \,)^T \tag{2-67b}$$

and

$$\mathbf{A}^{(\mathrm{TE})}(s) = \begin{pmatrix} 0 & -j(\epsilon_2 - \kappa^2) \\ -js^2 & 0 \end{pmatrix} \tag{2-67c}$$

Eqs. (2-65), (2-66) and (2-67) can be solved by several different method depending upon the choice of permittivity profiles in the core. For the case of a step-index fiber, either isotropic or uniaxial, an analytic solution of eq. (2-65) in terms of Bessel functions [16], [18] is possible. This analytic solution is identical to the exact solutions given in section 2.2.3. For the case of an isotropic graded-index fiber an approximate method using the concept of transition matrices [16] can be used.

For the more general cases of a uniaxial or biaxial graded-index fiber these two previous methods are not applicable. The first method can be used in an approximate manner for an uniaxial graded-index fiber by assuming the permittivities are piecewise continuous. This is equivalent to the stratification technique which will be discussed in section 3. The second method can not be used for either a uniaxial or biaxial graded-index fiber because the formulation depends upon a symmetry in the matrix $\mathbf{A}(s)$ which is present only for the

isotropic case.

What is needed is a solution method which can be used with all possible types of fibers, isotropic, uniaxial and biaxial, step or graded-index. One such method is the method of partitioning of systems of equations [22]. This method involves transforming a system of first order linear differential equations into a system of equations whose solutions are easier to find. The solution obtained using this method is valid wherever the Taylor series expansion for $\mathbf{A}(s)$ is valid. The form of the solution method presented in the following section is based on the expansion of $\mathbf{A}(x)$ in positive powers of x in contrast to the usual form where the expansion is in terms of positive powers of $1/x$.

The reason for using this alternative formulation should now be readily apparent. If the relative permittivities are of the form given by eq. (2-44) the poles of $\mathbf{A}(s)$ are located outside the fiber core in the region $r > 1$. The series expansion is therefore valid for the entire fiber core. In contrast the system obtained by writing eq. (2-17) in matrix form has poles in the region $0 < r < 1$ whenever either $\epsilon_1(r)$ or $\epsilon_2(r)$ is not a constant.

2.3.2 Series Expansion

Consider the following system of n linear differential equations

$$\frac{d\mathbf{u}}{dx} = \frac{1}{x^q}\mathbf{A}(x)\mathbf{u}(x), \quad \text{as } x \to 0 \tag{2 - 68}$$

where $\mathbf{u}$ is a column vector, q is an integer greater than or equal to 1 and $\mathbf{A}$ is a $N \times N$ matrix given by

$$\mathbf{A}(x) = \sum_{n=0}^{\infty} \mathbf{A}_n x^n \quad \text{as } x \to 0. \tag{2 - 69}$$

We seek formal solutions of the form

$$\mathbf{u}(x) = \mathbf{y}(x) x^{\sigma} e^{\Lambda(x)} \tag{2 - 70}$$

where σ is a constant,

$$\Lambda(x) = \sum_{n=1}^{q+1} -\frac{\lambda_n}{m} x^n \tag{2 - 71}$$

with $\lambda_{-n} = 0$ for $n \geq 0$ and

$$\mathbf{y}(x) = \sum_{n=0}^{\infty} \mathbf{y}_n x^n \quad \text{as } x \to 0 \tag{2 - 72}$$

Substituting eqs. (2-69) to (2-72) into eq. (2-68) and equating powers of x, we obtain equations to determine successively λ_n, σ and $\mathbf{y}_n$.

2.3.3 Asymptotic Partitioning of Systems of Equations

It is possible to simplify the system of equations by transforming them into some special differential equations whose solutions are easier to find. Let

$$\mathbf{u}(x) = \mathbf{P}(x)\mathbf{v}(x) \tag{2-73}$$

where $\mathbf{u}$ and $\mathbf{v}$ are column vectors and $\mathbf{P}(x)$ is a $N \times N$ nonsingular matrix. Using eq. (2-73), eq. (2-68) can be transformed into

$$\frac{d\mathbf{v}}{dx} = \frac{1}{x^q}\mathbf{B}(x)\mathbf{v}(x) \tag{2-74}$$

where

$$\mathbf{B}(x) = \mathbf{P}(x)^{-1}\left[\mathbf{A}(x)\mathbf{P}(x) - x^q\frac{d\mathbf{P}(x)}{dx}\right] \tag{2-75}$$

or

$$x^q\frac{d\mathbf{P}(x)}{dx} = \mathbf{A}(x)\mathbf{P}(x) - \mathbf{P}(x)\mathbf{B}(x). \tag{2-76}$$

Choose $\mathbf{P}(x)$ such that $\mathbf{B}(x)$ has a Jordan canonical form. To do this, let

$$\begin{aligned}\mathbf{B}(x) &= \sum_{n=0}^{\infty}\mathbf{B}_n x^n \quad \text{as } x \to 0,\\ \mathbf{P}(x) &= \sum_{n=0}^{\infty}\mathbf{P}_n x^m \quad \text{as } x \to 0,\end{aligned} \tag{2-77}$$

where $\mathbf{B}_n$ represents a Jordan canonical matrix. The left hand side of eq. (2-76) can then be written as

$$x^q\frac{d\mathbf{P}(x)}{dx} = \sum_{n=1}^{\infty} n\mathbf{P}_m x^{n+q-1} = \sum_{n=q}^{\infty}(n-q+1)\mathbf{P}_{n-q+1}x^n \tag{2-78}$$

while the right hand side of eq. (2-76) can be written as

$$\mathbf{A}(x)\mathbf{P}(x) - \mathbf{P}(x)\mathbf{B}(x) = \sum_{n=0}^{\infty}\left[\sum_{l=0}^{n}(\mathbf{A}_l\mathbf{P}_{n-l} - \mathbf{P}_l\mathbf{B}_{n-l})\right]x^n. \tag{2-79}$$

Using eqs. (2-78) and (2-77) eq. (2-76) can be written as

$$\sum_{n=q}^{\infty}(n-q+1)\mathbf{P}_{n-q+1}x^n = \sum_{n=0}^{\infty}\left[\sum_{l=0}^{n}(\mathbf{A}_l\mathbf{P}_{n-l} - \mathbf{P}_l\mathbf{B}_{n-l})\right]x^n \qquad (2-80)$$

Equating like powers of x we obtain

$$\mathbf{A}_0\mathbf{P}_0 - \mathbf{P}_0\mathbf{B}_0 = 0 \qquad (2-81)$$

for x^0 and for x^n, $n \geq 1$,

$$(n-q+1)\mathbf{P}_{n-q+1} = \sum_{l=0}^{n}(\mathbf{A}_l\mathbf{P}_{n-l} - \mathbf{P}_l\mathbf{B}_{n-l}) \qquad (2-82)$$

where $\mathbf{P}_{n-q+1} = 0$ for $n - q + 1 < 0$. Rewrite eqs. (2-81) and (2-82) as

$$\mathbf{B}_0 = \mathbf{P}_0^{-1}\mathbf{A}_0\mathbf{P}_0 \qquad (2-83)$$

and

$$\mathbf{A}_0\mathbf{P}_n - \mathbf{P}_n\mathbf{B}_0 = (n-q+1)\mathbf{P}_{n-q+1} - \sum_{l=0}^{n-1}(\mathbf{A}_{n-l}\mathbf{P}_l - \mathbf{P}_n\mathbf{B}_{n-l}) \qquad (2-84)$$

where $\mathbf{P}_0$ is chosen so that $\mathbf{B}_0$ is a Jordan canonical matrix. Multiplying eq. (2-84) from the left by $\mathbf{P}_0^{-1}$ and pulling the first term out of the summation gives

$$\begin{aligned}\mathbf{P}_0^{-1}\mathbf{A}_0\mathbf{P}_n - \mathbf{P}_0^{-1}\mathbf{P}_n\mathbf{B}_0 &= \\ &= (n-q+1)\mathbf{P}_0^{-1}\mathbf{P}_{n-q+1} - \mathbf{P}_0^{-1}\mathbf{A}_n\mathbf{P}_0 + \mathbf{P}_0^{-1}\mathbf{P}_0\mathbf{B}_n \\ &\quad - \mathbf{P}_0^{-1}\sum_{l=1}^{n-1}(\mathbf{A}_{n-l}\mathbf{P}_l - \mathbf{P}_l\mathbf{B}_{n-l})\end{aligned} \qquad (2-85)$$

Now define the matrices $\mathbf{W}_n$ and $\mathbf{F}_n$ as

$$\mathbf{W}_n = \mathbf{P}_0^{-1}\mathbf{P}_n \qquad (2-86)$$

and

$$\mathbf{F}_n = \mathbf{P}_0^{-1}\mathbf{A}_n\mathbf{P}_0 + \mathbf{P}_0^{-1}\sum_{l=1}^{n-1}(\mathbf{A}_{n-l}\mathbf{P}_l - \mathbf{P}_l\mathbf{B}_{n-l}). \qquad (2-87)$$

Eq. (2-85) can then be written as

$$\mathbf{B}_0\mathbf{W}_n - \mathbf{W}_n\mathbf{B}_0 = (n - q + 1)\mathbf{W}_{n-q+1} + \mathbf{B}_n - \mathbf{F}_n \qquad (2-88)$$

If $\mathbf{P}_0$ can be chosen so that $\mathbf{B}_0$ is diagonal then it can be seen from eq. (2-16) that

$$(\mathbf{B}_0)_{ii} = \lambda_i \quad i = 1, 2, \ldots, N \qquad (2-89)$$

where λ_i is the i'th eigenvalue of $\mathbf{A}_0$. When $\mathbf{B}_0$ is a diagonal matrix the expression $\mathbf{B}_0\mathbf{W}_n - \mathbf{W}_n\mathbf{B}_0$ has zeroes along its main diagonal. Eq. (2-88) can be easily satisfied by setting

$$(\mathbf{B}_n)_{ij} = \begin{cases} (\mathbf{F}_n)_{ii}, & i = j; \\ 0, & i \neq j, \end{cases} \qquad (2-89a)$$

and

$$(\mathbf{W}_n)_{ii} = 0 \quad \text{for } n > 0. \qquad (2-89b)$$

For the particular case $q = 1$ eq. (2-88) reduces to

$$(\mathbf{B}_0 - n\mathbf{I})\mathbf{W}_n - \mathbf{W}_n\mathbf{B}_0 = \mathbf{B}_n - \mathbf{F}_n \qquad (2-91)$$

where, using $w_n^{ij} = (\mathbf{W}_n)_{ij}$ and $f_n^{ij} = (\mathbf{F}_n)_{ij}$,

$$(\mathbf{B}_0 - n\mathbf{I}) - \mathbf{W}_n\mathbf{B}_0 = \begin{pmatrix} 0 & (\lambda_1 - \lambda_2 - n)w_n^{12} & (\lambda_1 - \lambda_3 - n)w_n^{13} & (\lambda_1 - \lambda_3 - n)w_n^{14} \\ (\lambda_2 - \lambda_1 - n)w_n^{21} & 0 & (\lambda_2 - \lambda_3 - n)w_n^{23} & (\lambda_2 - \lambda_3 - n)w_n^{24} \\ (\lambda_3 - \lambda_1 - n)w_n^{31} & (\lambda_3 - \lambda_2 - n)w_n^{32} & 0 & (\lambda_3 - \lambda_3 - n)w_n^{34} \\ (\lambda_3 - \lambda_1 - n)w_n^{41} & (\lambda_3 - \lambda_2 - n)w_n^{42} & (\lambda_3 - \lambda_3 - n)w_n^{43} & 0 \end{pmatrix} \qquad (2-92)$$

and

$$\mathbf{B}_n - \mathbf{F}_n = -\begin{pmatrix} 0 & f_n^{12} & f_n^{13} & f_n^{14} \\ f_n^{21} & 0 & f_n^{23} & f_n^{24} \\ f_n^{31} & f_n^{32} & 0 & f_n^{34} \\ f_n^{41} & f_n^{42} & f_n^{43} & 0 \end{pmatrix}. \qquad (2-93)$$

Eqs. (2-91), (2-92) and (2-93) can be used to find $\mathbf{W}_n$. Note, if $\lambda_i - \lambda_j - n = 0$ and $f_n^{ij} \neq 0$ it may not be possible to find $\mathbf{W}_n$ and therefore it may not be possible to find a solution.

Using eqs. (2-86) to (2-90) the matrices $\mathbf{B}_n$ and $\mathbf{P}_n$, $n = 1, 2, 3, \ldots$ can be found using an iterative procedure. After completing the desired number of iterations the matrices $\mathbf{B}(x)$ and $\mathbf{P}(x)$ can be approximated by series constructed from $\mathbf{B}_n$ and $\mathbf{P}_n$, $n = 1, 2, 3, \ldots$. Since $\mathbf{B}(x)$ is a Jordan canonical matrix eq. (2-73) can be easily solved for the elements of the vector $\mathbf{v}(x)$. Then, using eq. (2-73) the solution for the vector $\mathbf{u}$ in the original problem can be found.

As an example of this solution method let us consider Bessel's equation

$$x^2 \frac{d^2 y}{dx^2} + x\frac{dy}{dx} + (x^2 - m^2)y = 0 \qquad (2-94)$$

or equivalently

$$x\frac{d}{dx}\left(x\frac{dy}{dx}\right) + (x^2 - m^2)y = 0\,. \qquad (2-95)$$

Letting

$$y = u_1 \qquad \text{and} \qquad x\frac{dy}{dx} = u_2 \qquad (2-96)$$

eq. (2-95) is transformed into

$$\begin{pmatrix} x\frac{du_1}{dx} \\ x\frac{du_2}{dx} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ m^2 - x^2 & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \qquad (2-97)$$

or equivalently

$$\frac{d\mathbf{u}}{dx} = \frac{1}{x}\mathbf{A}(x)\mathbf{u} \qquad (2-98)$$

where

$$\mathbf{A}(x) = \begin{pmatrix} 0 & 1 \\ m^2 - x^2 & 0 \end{pmatrix}. \qquad (2-99)$$

Comparing eq. (2-98) with eq. (2-68) we see immediately that $q = 1$. From eq. (2-99) we have $\mathbf{A}_n = 0$ for $n > 2$ and

$$\mathbf{A}_0 = \begin{pmatrix} 0 & 1 \\ m^2 & 0 \end{pmatrix}, \quad \mathbf{A}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix}. \tag{2-100}$$

If $m \neq 0$ the eigenvalues of $\mathbf{A}_0$ are $\pm m$ and the matrices $\mathbf{P}_0$ and $\mathbf{P}_0^{-1}$ can be chosen as

$$\mathbf{P}_0 = \begin{pmatrix} 1 & 1 \\ m & -m \end{pmatrix} \qquad \text{and} \qquad \mathbf{P}_0^{-1} = \frac{1}{2m} \begin{pmatrix} m & 1 \\ m & -1 \end{pmatrix} \tag{2-101}$$

so that

$$\mathbf{B}_0 = \mathbf{P}_0^{-1}\mathbf{A}_0\mathbf{P}_0 = \begin{pmatrix} m & 0 \\ 0 & -m \end{pmatrix}. \tag{2-102}$$

From eq. (2-87) we have

$$\mathbf{F}_1 = \mathbf{P}_0^{-1}\mathbf{A}_1\mathbf{P}_0 = 0 \tag{2-103}$$

and therefore $\mathbf{B}_1 = \mathbf{W}_1 = \mathbf{P}_1 = 0$. Since $\mathbf{B}_1$,$\mathbf{W}_1$ and $\mathbf{P}_1$ are all identically equal to zero, from eq. (2-87)

$$\mathbf{F}_2 = \mathbf{P}_0^{-1}\mathbf{A}_2\mathbf{P}_0 = \frac{1}{2m} \begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix}, \tag{2-104}$$

from eq. (2-90)

$$\mathbf{B}_2 = \frac{1}{2m} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \tag{2-105}$$

and from eqs. (2-91), (2-92) and (2-93)

$$\mathbf{W}_2 = \frac{1}{4m} \begin{pmatrix} 0 & \frac{1}{m-1} \\ \frac{1}{m+1} & 0 \end{pmatrix} \tag{2-106}$$

Also, from eq. (2-86)

$$\mathbf{P}_2 = \mathbf{P}_0\mathbf{W}_2 = \frac{1}{4m} \begin{pmatrix} \frac{1}{m+1} & \frac{1}{m-1} \\ -\frac{m}{m+1} & \frac{m}{m+1} \end{pmatrix} \tag{2-107}$$

Notice that both $\mathbf{W}_2$ and $\mathbf{P}_2$ are undefined when $m = \pm 1$ and as mentioned earlier a solution may not be possible. For the moment ignore this problem with $\mathbf{W}_2$ and $\mathbf{P}_2$ and

proceed with the solution. It will be shown later that this problem can be alleviated by the appropriate choice of integration constants.

Continuing the solution procedure we find $\mathbf{F}_3$, $\mathbf{B}_3$, $\mathbf{W}_3$ and $\mathbf{P}_3$ are all identically equal to zero. The fourth iteration of the procedure gives

$$\mathbf{F}_4 = \frac{1}{8m^2}\begin{pmatrix} -\frac{1}{m+1} & -\frac{2}{m-1} \\ \frac{2}{m+1} & \frac{1}{m-1} \end{pmatrix} \tag{2-108a}$$

$$\mathbf{B}_4 = \frac{1}{8m^2}\begin{pmatrix} -\frac{1}{m+1} & 0 \\ 0 & \frac{1}{m-1} \end{pmatrix} \tag{2-108b}$$

$$\mathbf{W}_4 = \frac{1}{8m^2}\begin{pmatrix} 0 & \frac{1}{(m-1)(m-2)} \\ \frac{1}{(m+1)(m+2)} & 0 \end{pmatrix} \tag{2-108c}$$

and

$$\mathbf{P}_4 = \frac{1}{8m^2}\begin{pmatrix} \frac{1}{(m+1)(m+2)} & \frac{1}{(m-1)(m-2)} \\ \frac{-m}{(m+1)(m+2)} & \frac{m}{(m-1)(m-2)} \end{pmatrix}. \tag{2-108d}$$

The matrices $\mathbf{B}(x)$ and $\mathbf{P}(x)$ can be approximated as

$$\mathbf{B}(x) \approx \mathbf{B}_0 + \mathbf{B}_2 x^2 + \mathbf{B}_4 x^4 \tag{2-109a}$$

$$\mathbf{P}(x) \approx \mathbf{P}_0 + \mathbf{P}_2 x^2 + \mathbf{P}_4 x^4. \tag{2-109b}$$

Substituting eq. (2-109a) for $\mathbf{B}(x)$ in eq.(2-74) gives

$$\begin{aligned} \frac{dv_1}{dx} &= \frac{1}{x}\Big[(\mathbf{B}_0)_{11} + (\mathbf{B}_2)_{11}x^2 + (\mathbf{B}_4)_{11}x^4\Big]v_1 \\ &= \frac{1}{x}\left[m - \frac{x^2}{2m} - \frac{x^4}{8m^2(m+1)}\right]v_1 \end{aligned} \tag{2-110a}$$

and

$$\begin{aligned} \frac{dv_2}{dx} &= \frac{1}{x}\Big[(\mathbf{B}_0)_{22} + (\mathbf{B}_2)_{22}x^2 + (\mathbf{B}_4)_{22}x^4\Big]v_2 \\ &= \frac{1}{x}\left[-m + \frac{x^2}{2m} + \frac{x^4}{8m^2(m-1)}\right]v_2 \end{aligned} \tag{2-110b}$$

Solving eq. (2-110) for v_1 and v_2 gives

$$v_1(x) = C_1 x^m e^{-\frac{x^2}{4m}\left[1+\frac{x^2}{8m(m+1)}\right]} \qquad (2-111a)$$

and

$$v_2(x) = C_2 x^{-m} e^{\frac{x^2}{4m}\left[1+\frac{x^2}{8m(m-1)}\right]} \qquad (2-111b)$$

where C_1 and C_2 are integration constants and $m \neq 0$.

Notice that $v_1(x)$ is undefined when $m = -1$ and $v_2(x)$ is undefined when $m = 1$. Also notice that the matrices $\mathbf{P}_2$ and $\mathbf{P}_4$ are undefined when $m = \pm 1$ and in addition $\mathbf{P}_4$ is undefined when $m = \pm 2$. In fact if more iterations are performed one would find that the matrix $\mathbf{P}_{2k}$ is undefined when $m = \pm 1, \pm 2, \ldots, \pm k$. It appears the eventual solution for $u_1(x)$ and $u_2(x)$ will always be undefined when m is an integer in the interval $[-k \ldots k]$ where $2k$ is the number of iterations performed. This problem can be easily overcome by setting $C_2 = 0$ when $m > 0$ and $C_1 = 0$ when $m < 0$. However, the two solutions which are obtained are not independent solutions when m is an integer. A careful inspection of the expressions for $v_1(x)$, $v_2(x)$, $\mathbf{P}_2$ and $\mathbf{P}_4$ show that

$$v_2(x)\Big|_{m=-m} = v_1(x) \qquad (2-112a)$$

$$(\mathbf{P}_2)_{i2}\Big|_{m=-m} = (P_2)_{i1} \qquad (2-112b)$$

and

$$(\mathbf{P}_4)_{i2}\Big|_{m=-m} = (P_4)_{i1} \qquad (2-112c)$$

where $i = 1, 2$ and $(\mathbf{P}_n)_{ij}$ is an element of $\mathbf{P}_n$. Therefore, it is only necessary to consider the solutions for $m > 0$.

We can then write

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} (\mathbf{P}_0)_{11} + (\mathbf{P}_2)_{11}x^2 + (\mathbf{P}_4)_{11}x^4 \\ (\mathbf{P}_0)_{21} + (\mathbf{P}_2)_{21}x^2 + (\mathbf{P}_4)_{21}x^4 \end{pmatrix} v_1$$

$$= C_1 x^m \begin{pmatrix} 1 + \frac{x^2}{4m(m+1)}\left[1 + \frac{x^2}{2m(m+2)}\right] \\ m - \frac{x^2}{4(m+1)}\left[1 + \frac{x^2}{2m(m+2)}\right] \end{pmatrix} e^{-\frac{x^2}{4m}\left[1+\frac{x^2}{8m(m+1)}\right]} \qquad (2-113)$$

The constant C_1 can be determined by examining the solution of u_2 at $x = 0$ when $m = 1$.

When $m = 1$, $u_2(x) = x\, dJ_1/dx$ or

$$\frac{dJ_1(x)}{dx} = \frac{u_2(x)}{x} = C_1\left[1 - \frac{x^2}{8}\left(1 + \frac{x^2}{6}\right)\right] e^{-\frac{x^2}{4}\left(1+\frac{x^2}{16}\right)}$$

but at $x = 0$, $dJ_1/dx = 1/2$ and $u_2/x = C_1$ therefore $C_1 = 1/2$. The solution of eq. (2-94) for $m > 0$ can be written as

$$y(x) = \frac{1}{2}x^m\left\{1 + \frac{x^2}{4m(m+1)}\left[1 + \frac{x^2}{2m(m+2)}\right]\right\} e^{-\frac{x^2}{4m}\left[1+\frac{x^2}{8m(m+1)}\right]} . \qquad (2-114)$$

Now consider the solution of eq. (2-94) when $m = 0$. From eqs. (2-98) and (2-99) we have $q = 1$, $\mathbf{A}_n = 0$ for $n > 2$,

$$\mathbf{A}_0 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{A}_1 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} \qquad (2-115)$$

and $\mathbf{A}_0$ has two eigenvalues equal to zero. Since $\mathbf{A}_0$ is already in Jordan canonical form let $\mathbf{P}_0 = \mathbf{P}_0^{-1} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix, so that

$$\mathbf{B}_0 = \mathbf{P}_0^{-1}\mathbf{A}_0\mathbf{P}_0 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad (2-116)$$

is also in Jordan canonical form. Performing four iterations of the solution procedure results

in the following matrices:

$$
\begin{aligned}
&\mathbf{F}_1 = \mathbf{B}_1 = \mathbf{W}_1 = \mathbf{P}_1 = 0 \\
&\mathbf{F}_2 = \mathbf{P}_0^{-1}\mathbf{A}_2\mathbf{P}_0 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix} \\
&\mathbf{B}_2 = 0 \\
&\mathbf{W}_2 = \mathbf{P}_2 = \frac{1}{4}\begin{pmatrix} -1 & 1 \\ -2 & 1 \end{pmatrix} \\
&\mathbf{F}_3 = \mathbf{B}_3 = \mathbf{W}_3 = \mathbf{P}_3 = 0 \qquad\qquad\qquad (2-117) \\
&\mathbf{F}_4 = \mathbf{P}_0^{-1}\mathbf{A}_2\mathbf{P}_0\frac{1}{4}\begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix} \\
&\mathbf{B}_4 = \frac{1}{4}\begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix} \\
&\mathbf{W}_4 = \mathbf{P}_4 = \frac{1}{128}\begin{pmatrix} 2 & -1 \\ 8 & -2 \end{pmatrix}
\end{aligned}
$$

The matrix $\mathbf{B}(x)$ can be written as

$$\mathbf{B}(x) \approx \mathbf{B}_0 + \mathbf{B}_4 x^4 = \begin{pmatrix} 0 & 1 \\ 0 & -\frac{x^4}{4} \end{pmatrix} \qquad (2-117)$$

Eq. (2-74) can then be written as

$$\frac{dv_1}{dx} = \frac{1}{x}v_2 \qquad (2-118a)$$

and

$$\frac{dv_2}{dx} = -\frac{x^4}{4}v_2 \qquad (2-118b)$$

Solving eq. (2-118) first for v_2 and then for v_1 results in

$$v_2(x) = C_1 e^{-x^4/16} \qquad (2-119a)$$

and

$$
\begin{aligned}
v_1(x) &= \int \frac{C_1}{x} e^{-x^4/16}\, dx \\
&= C_2 + C_1 e^{-x^4/16}\ln(x) + C_1 \int \frac{x^3}{4} e^{-x^4/16}\ln(x)\, dx \qquad (2-119b)
\end{aligned}
$$

where C_1 and C_2 are integration constants. In order for $v_1(x)$ to be finite at $x = 0$ (assumes the desired solution is J_0 not K_0) the constant C_1 must be identically equal to zero,.or $v_1(x) = C_2$ and $v_2(x) = 0$. The solutions of eq. (2-98) can then be written as

$$\begin{aligned} \mathbf{u}(x) &= \mathbf{P}(x)\mathbf{v}(x) \\ &= C_2 \begin{pmatrix} 1 - \frac{x^2}{4} + \frac{x^4}{64} \\ -\frac{x^2}{2} + \frac{x^4}{16} \end{pmatrix} \end{aligned} \tag{2-120}$$

When $m = 0$ the solution of eq. (2-94) can be written as

$$y(x) = C_2 \left(1 - \frac{x^2}{4} + \frac{x^4}{64}\right) \tag{2-121}$$

which is simply the truncated series expansion for $J_0(x)$.

2.3.4 Solutions for Transverse Modes

Assuming the individual elements of the matrices $\mathbf{A}^{(TE)}(s)$ and $\mathbf{A}^{(TM)}(s)$ can be expanded as Taylor series, a completely general solution to eqs. (2-66) and (2-67) can be found in terms of the coefficients from the series expansions. After two or three iterations of the solution procedure the resulting matrices become cumbersome and further iterations are tedious. If the form of the permittivity profiles is known in advance the iteration procedure can often be made more manageable.

Let us assume the permittivity profiles are of the form given by eq. (2-44). In particular choose all the profiles to have a parabolic shape i.e. $\alpha_i = 2 \quad i = 1,2,3$. It should be noted that since $\epsilon_1(r)$ and $\epsilon_2(r)$ must be equal at $r = 0$, it is necessary for $\epsilon_1(0) = \epsilon_2(0)$ and $\Delta_1 = \Delta_2$. Since $\alpha_1 = \alpha_2 = 2$ this choice for the permittivity profiles does not strictly contain an example of a biaxial graded-index fiber. If however, in the final result either Δ_1 or Δ_2 but not both is set to zero then the resulting solution is a valid example of a biaxial

graded-index fiber where either $\epsilon_1(r)$ has a parabolic profile and $\epsilon_2(r)$ is a constant or $\epsilon_1(r)$ is a constant and $\epsilon_2(r)$ has a parabolic profile.

If all three permittivities have parabolic profiles then

$$\epsilon_i(s) = \epsilon_i(1 - 2\Delta_i^0 s^2) \qquad i = 1,2,3 \tag{2-122}$$

where $s = k_0\rho = k_0 ar$ and $\Delta_i^0 = \Delta_i/(k_0 a)^2$. Substituting eq. (2-122) into the expression for $\mathbf{A}^{(\mathrm{TM})}$ given by eq. (2-66c) and expanding $\mathbf{A}^{(\mathrm{TM})}$ as a series results in $\mathbf{A}_{2n-1}^{(\mathrm{TM})} = 0$ for $n = 1,2,3,\ldots$ and

$$\begin{aligned}
\mathbf{A}_0^{(\mathrm{TM})} &= \begin{pmatrix} 0 & \frac{j}{\epsilon_1}k_{N1}^2 \\ 0 & 0 \end{pmatrix} \\
\mathbf{A}_2^{(\mathrm{TM})} &= \begin{pmatrix} 0 & -\frac{j}{\epsilon_1}(2\Delta_1^0)\kappa^2 \\ j\epsilon_3 & 0 \end{pmatrix} \\
\mathbf{A}_4^{(\mathrm{TM})} &= \begin{pmatrix} 0 & -\frac{j}{\epsilon_1}(2\Delta_1^0)^2\kappa^2 \\ -j2\epsilon_3\Delta_3^0 & 0 \end{pmatrix} \\
\mathbf{A}_{2n}^{(\mathrm{TM})} &= \begin{pmatrix} 0 & -\frac{j}{\epsilon_1}(2\Delta_1^0)^n\kappa^2 \\ 0 & 0 \end{pmatrix}
\end{aligned} \tag{2-123}$$

for $n = 1,2,3,\ldots$ where $k_{N1}^2 = \epsilon_1 - \kappa^2$. Similarly the expansion of $\mathbf{A}^{(\mathrm{TE})}$ gives $\mathbf{A}_n^{(\mathrm{TE})} = 0$ for $n = 1$ and $n > 2$,

$$\mathbf{A}_0^{(\mathrm{TE})} = \begin{pmatrix} 0 & -jk_{N2}^2 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 0 & j2\epsilon_2\Delta_2^0 \\ -j & 0 \end{pmatrix} \tag{2-124}$$

where $k_{N2}^2 = \epsilon_2 - \kappa^2$. Since the two eigenvalues of both $\mathbf{A}_0^{(\mathrm{TM})}$ and $\mathbf{A}_0^{(\mathrm{TE})}$ are identically equal to zero the matrix $\mathbf{P}_0$ in both cases must be chosen so that $\mathbf{B}_0$ is Jordan canonical matrix. Since $\mathbf{A}_0^{(\mathrm{TM})}$ and $\mathbf{A}_0^{(\mathrm{TE})}$ are of the form

$$\mathbf{A}_0 = \begin{pmatrix} 0 & \tau \\ 0 & 0 \end{pmatrix} \tag{2-125}$$

if $\mathbf{P}_0$ is chosen as

$$\mathbf{P}_0 = \begin{pmatrix} \tau & 0 \\ 0 & 1 \end{pmatrix} \tag{2-126}$$

then

$$\mathbf{B}_0 = \mathbf{P}_0^{-1}\mathbf{A}_0\mathbf{P}_0 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \qquad (2-127)$$

where eq. (2-127) is valid for both $\mathbf{A}_0^{(\mathrm{TM})}$ and $\mathbf{A}_0^{(\mathrm{TE})}$.

Since $\mathbf{B}_0$ is not a diagonal matrix the choices made in eq. (2-89b) for the elements of $\mathbf{W}_n$ will not in general satisfy eq. (2-88). For the particular case $q = 1$ if the elements of $\mathbf{B}_n$ are chosen using eq. (2-89a) the elements of $\mathbf{W}_n$ must be chosen so that eq. (2-91) is satisfied. For a 2×2 matrix eq. (2-91) can be written as

$$\begin{pmatrix} w_n^{21} - nw_n^{11} & w_n^{22} - nw_n^{12} - w_n^{11} \\ -nw_n^{21} & -nw_n^{22} - w_n^{21} \end{pmatrix} = \begin{pmatrix} 0 & -f_n^{12} \\ -f_n^{21} & 0 \end{pmatrix} \qquad (2-128)$$

where $w_n^{ij} = (\mathbf{W}_n)_{ij}$ and $f_n^{ij} = (\mathbf{F}_n)_{ij}$. Solving eq. (2-128) for the elements of $\mathbf{W}_n$ gives

$$\begin{aligned} w_n^{21} &= \frac{f_n^{21}}{n} \\ w_n^{11} &= \frac{w_n^{21}}{n} = \frac{f_n^{21}}{n^2} \\ w_n^{22} &= -\frac{w_n^{21}}{n} = \frac{-f_n^{21}}{n^2} \\ w_n^{12} &= \frac{w_n^{22} - w_n^{11} + f_n^{12}}{n} = \frac{n^2 f_n^{12} - 2f_n^{21}}{n^3} \end{aligned} \qquad (2-129)$$

Eqs. (2-66) and (2-67) can now be solved iteratively using eqs. (2-86), (2-87), (2-89a) and (2-129).

After performing N iterations the matrix $\mathbf{B}(s)$ for either eq. (2-66) or (2-67) can be written in the form

$$\mathbf{B}(s) = \begin{pmatrix} B_{11}(s) & 1 \\ 0 & B_{22}(s) \end{pmatrix} \qquad (2-130a)$$

where

$$B_{ii}(s) = \sum_{n=1}^{N} (\mathbf{B}_n)_{ii} s^n \qquad i = 1, 2\,. \qquad (2-130b)$$

Note that the summation in eq. (2-130b) starts at $n = 1$ because $(\mathbf{B}_n)_{ii} = 0 \quad i = 1,2$. Substituting eq. (2-130) into eq. (2-74) gives

$$\frac{dv_1}{ds} = \frac{1}{s}[B_{11}(s)v_1 + v_2] \tag{2-131a}$$

$$\frac{dv_2}{ds} = \frac{1}{s}B_{22}(s)v_2\,. \tag{2-131b}$$

Solving eq. (2-131) first for $v_2(s)$ then for $v_1(s)$ gives

$$v_2(s) = C_2 e^{\lambda_2(s)} \tag{2-132a}$$

$$v_1(s) = C_1 e^{\lambda_1(s)} + \int \frac{C_2}{s} e^{\lambda_2(s)-\lambda_1(s)}\,ds \tag{2-132b}$$

where

$$\begin{aligned} \lambda_i(s) &= \int \frac{1}{s} B_{ii}(s)\,ds \qquad i = 1,2 \\ &= \sum_{n=1}^{N} (\mathbf{B}_n)_{ii} \frac{s^n}{n} \qquad i = 1,2\,. \end{aligned} \tag{2-132c}$$

In order for $v_1(s)$ to be finite at $s = 0$ it is necessary to choose C_2 to be identically equal to zero in eq. (2-132). The solutions for $v_1(s)$ and $v_2(s)$ can then be written as

$$v_1(s) = C_1 e^{\lambda_1(s)} \tag{2-133a}$$

$$v_2(s) = 0\,. \tag{2-133b}$$

The solution to the original problem now is written as

$$\begin{aligned} \mathbf{u}(s) &= \mathbf{P}(s)\mathbf{v}(s) \\ &= C_1 \begin{pmatrix} P_{11}(s) \\ P_{21}(s) \end{pmatrix} e^{\lambda_1(s)} \end{aligned} \tag{2-134a}$$

where

$$P_{ij}(s) = \sum_{n=0}^{N} (\mathbf{P}_n)_{ij} s^n \,. \tag{2-134b}$$

If four iterations are performed for the transverse magnetic case using the $\mathbf{A}_n^{(\mathrm{TM})}$'s given in eq. (2-123) the solutions for E_z and sh_ϕ can be written as

$$E_z = \frac{j}{\epsilon_3} k_{N1}^2 C_1 \left\{ 1 - \frac{\epsilon_3}{\epsilon_1} \frac{k_{N1}^2}{4} s^2 + \left[\left(\frac{\epsilon_3}{\epsilon_1}\right)^2 \frac{k_{N1}^4}{64} + \frac{\epsilon_3}{\epsilon_1} \Delta_3^0 \frac{k_{N1}^2}{8} \right] s^4 \right\} e^{\frac{\epsilon_3}{\epsilon_1}(\Delta_1^0 \kappa^2)\frac{s^4}{4}} \tag{2-135a}$$

$$sh_\phi = -C_1 \left\{ \frac{\epsilon_3}{\epsilon_1} \frac{k_{N1}^2}{2} s^2 + \left[\left(\frac{\epsilon_3}{\epsilon_1}\right)^2 \frac{k_{N1}^4}{16} + \frac{\epsilon_3}{\epsilon_1} \Delta_3^0 \frac{k_{N1}^2}{2} \right] s^4 \right\} e^{\frac{\epsilon_3}{\epsilon_1}(\Delta_1^0 \kappa^2)\frac{s^4}{4}} \,. \tag{2-135b}$$

Similarly for the transverse electric case h_z and sE_ϕ are found to be

$$h_z = -j k_{N2}^2 C_1 \left[1 - \frac{k_{N2}^2}{4} s^2 + \frac{k_{N2}^4}{64} s^2 \right] e^{\epsilon_2 \Delta_2^0 \frac{s^4}{4}} \tag{2-136a}$$

$$sE_\phi = -C_1 \left[\frac{k_{N2}^2}{2} s^2 - \frac{k_{N2}^4}{16} s^4 \right] e^{\epsilon_2 \Delta_2^0 \frac{s^4}{4}} \,. \tag{2-136b}$$

An important question to ask at this time is whether or not these asymptotic solutions correspond to any known solutions, preferably an exact solution. The only comparison which can be made with an exact solution is for the case of either an isotropic or a uniaxial step-index fiber. The asymptotic solutions for the transverse modes in a step-index fiber are obtained by setting $\Delta_i^0 = 0$, $i = 1, 2, 3$ and $\epsilon_2 = \epsilon_1$ in eqs. (2-135) and (2-136). For the transverse electric case the asymptotic solutions for h_z and sE_ϕ are given by

$$h_z = -j k_{N1}^2 C_1 \left(1 - \frac{k_{N1}^2}{4} s^2 + \frac{k_{N1}^4}{64} s^4 \right) \tag{2-137a}$$

$$sE_\phi = C_1 \left(-\frac{k_{N1}^2}{2} s^2 + \frac{k_{N1}^4}{16} s^4 \right) \tag{2-137b}$$

and for the transverse magnetic case E_z and sh_ϕ are given by

$$E_z = \frac{j}{\epsilon_1} k_{N1}^2 C_1 \left[1 - \frac{\epsilon_3}{\epsilon_1} \frac{k_{N1}^2}{4} s^2 + \left(\frac{\epsilon_3}{\epsilon_1}\right)^2 \frac{k_{N1}^4}{64} s^4 \right] \tag{2-138a}$$

$$sh_\phi = C_1\left[-\frac{\epsilon_3}{\epsilon_1}\frac{k_{N1}^2}{2}s^2 + \left(\frac{\epsilon_3}{\epsilon_1}\right)^2\frac{k_{N1}^4}{16}s^4\right] \qquad (2-138b)$$

where $k_{N1}^2 = \epsilon_1 - \kappa^2$ for both eq. (2-137) and (2-138). From eqs. (2-8b), (2-8c) and (2-30) the exact solutions for the transverse electric case are given by

$$h_z = BJ_0(k_{N1}s) \qquad (2-139a)$$

$$sE_\phi = j\frac{s}{k_{N1}^2}\frac{dh_z}{ds} = -j\frac{s}{k_{N1}}BJ_1(k_{N1}s) \qquad (2-139b)$$

and for the transverse magnetic case the exact solution is given by

$$H_z = BJ_0\left(\sqrt{\frac{\epsilon_3}{\epsilon_1}}k_{N1}s\right) \qquad (2-140a)$$

$$sh_\phi = j\frac{s\epsilon_1}{k_{N1}^2}\frac{dE_z}{ds} = -j\frac{s\sqrt{\epsilon_1\epsilon_3}}{k_{N1}}BJ_1\left(\sqrt{\frac{\epsilon_3}{\epsilon_1}}k_{N1}s\right). \qquad (2-140b)$$

If $C_1 = jB/k_{N1}^2$ in eq. (2-137) and $C_1 = -j\epsilon_1 B/k_{N1}^2$ in eq. (2-138) then it is clear that eq. (2-137) and (2-138) are simply truncated series expansions for eqs. (2-139) and (2-140).

The allowable modes can be determined by solving the generalized dispersion relation given by eq. (2-25) For transverse modes the generalized dispersion relation separates into the following two equations

$$\frac{1}{\gamma a}\frac{K_m'(\gamma a)}{K_m(\gamma a)} + \frac{1}{(k_{t2}a)^2}\frac{h'}{h} = 0 \qquad (2-141a)$$

$$\frac{\epsilon_c}{\gamma a}\frac{K_m'(\gamma a)}{K_m(\gamma a)} + \frac{\epsilon_1}{(k_{t2}a)^2}\frac{e'}{e} = 0 \qquad (2-141b)$$

where e is the solution for E_z in the core evaluated at $r = 1$ ($s = k_0a$), $e' = de/dr$, h is the solution for h_z in the core evaluated at $r = 1$, $h' = dh/dr$, k_{t1} and k_{t2} are the transverse wavenumbers, eq. (2-9), evaluated at $r = 1$. Eq. (2-141a) is the dispersion relation for transverse electric modes and eq. (2-141b) is the dispersion relation for transverse magnetic

modes. For transverse electric modes the ratio h'/h is given by

$$\begin{aligned}\frac{h'}{h} &= \left.\frac{\frac{dh_z}{dr}}{h_z}\right|_{r=1} = \left.\frac{k_0 a \frac{dh_z}{ds}}{h_z}\right|_{s=k_0 a} = \left.-j\frac{k_{t2}^2(s)}{k_0^2}\frac{sE_\phi}{h_z}\right|_{s=k_0 a} \\ &= -j\frac{k_{t2}^2(k_0 a)}{k_0^2}\frac{P_{21}(k_0 a)}{P_{11}(k_0 a)} \qquad (2-142)\end{aligned}$$

where k_{t2}^2 is given by eq. (2-9). For transverse magnetic modes the ratio e'/e is given by

$$\begin{aligned}\frac{e'}{e} &= \left.\frac{\frac{dE_z}{dr}}{E_z}\right|_{r=1} = \left.\frac{k_0 a \frac{dE_z}{ds}}{E_z}\right|_{s=k_0 a} = \left.\frac{k_{t1}^2(s)}{k_0^2\epsilon_1(s)}\frac{sh_\phi}{E_z}\right|_{s=k_0 a} \\ &= \frac{k_{t1}^2(k_0 a)}{k_0^2\epsilon_1(k_0 a)}\frac{P_{21}(k_0 a)}{P_{11}(k_0 a)} \qquad (2-143)\end{aligned}$$

where k_{t1}^2 is given by eq. (2-9).

2.3.5 Solution for Hybrid Modes

As was the case for the solution of eqs. (2-66) and (2-67) for the transverse modes it is possible to generate a general solution for eq. (2-65) in terms of the elements of the coefficient matrices from the series expansion of $\mathbf{A}(s)$. In practice, however, it is not desirable to generate such a solution. Instead for mathematical convenience consider the solutions of eq. (2-65) for some particular permittivity profiles.

The two example which will be discussed are a biaxial graded-index fiber where $\epsilon_2(s)$ is a constant and a uniaxial graded-index fiber. In both cases $\epsilon_1(s)$ and $\epsilon_3(s)$ are chosen to have parabolic profiles. These two examples contain as special cases the solutions for a step-index fiber, either isotropic or uniaxial, and an isotropic graded-index fiber.

For the example of a biaxial graded-index fiber, if $\epsilon_1(s)$ and $\epsilon_3(s)$ are given by eq. (2-122)

the matrix $\mathbf{A}(s)$ can be expanded in terms of the following matrices

$$\mathbf{A}_0 = \begin{pmatrix} 0 & 0 & -j\frac{m\kappa}{\epsilon_1} & j\frac{k_{N1}^2}{\epsilon_1} \\ 0 & 0 & \frac{jm^2}{\epsilon_1} & j\frac{m\kappa}{\epsilon_1} \\ jm\kappa & -jk_{N1}^2 & 0 & 0 \\ -jm^2 & -jm\kappa & 0 & 0 \end{pmatrix} \qquad (2-144a)$$

$$\mathbf{A}_2 = \begin{pmatrix} 0 & 0 & -j\frac{m\kappa}{\epsilon_1}(2\Delta_1^0) & -j\frac{\kappa^2}{\epsilon_1}(2\Delta_1^0) \\ 0 & 0 & j\left[\frac{m^2}{\epsilon_1}(2\Delta_1^0) - 1\right] & j\frac{m\kappa}{\epsilon_1}(2\Delta_1^0) \\ 0 & 0 & 0 & 0 \\ j\epsilon_3 & 0 & 0 & 0 \end{pmatrix} \qquad (2-144b)$$

$$\mathbf{A}_4 = \begin{pmatrix} 0 & 0 & -j\frac{m\kappa}{\epsilon_1}(2\Delta_1^0)^2 & -j\frac{\kappa^2}{\epsilon_1}(2\Delta_1^0)^2 \\ 0 & 0 & \frac{jm^2}{\epsilon_1}(2\Delta_1^0)^2 & j\frac{m\kappa}{\epsilon_1}(2\Delta_1^0)^2 \\ 0 & 0 & 0 & 0 \\ -j\epsilon_3(2\Delta_3^0) & 0 & 0 & 0 \end{pmatrix} \qquad (2-144c)$$

$$\mathbf{A}_{2n} = \frac{(2\Delta_1^0)^n}{\epsilon_1}\begin{pmatrix} 0 & 0 & -jm\kappa & -j\kappa^2 \\ 0 & 0 & jm^2 & jm\kappa \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad n = 3,4,5,\ldots \qquad (2-144d)$$

and $\mathbf{A}_{2n-1} = 0$ for $n = 1,2,3,\ldots$ where $k_{N1}^2 = \epsilon_1 - \kappa^2$. Note, ϵ_2 does not appear in eq. (2-144) since $\epsilon_2(0) = \epsilon_1(0) = \epsilon_1$. For the example of a uniaxial graded-index fiber where $\epsilon_1(s)$ and $\epsilon_3(s)$ are again given by eq. (2-122) the matrix $\mathbf{A}(s)$ can be expanded as a series using

$$\mathbf{A}_0 = \begin{pmatrix} 0 & 0 & -j\frac{m\kappa}{\epsilon_1} & j\frac{k_{N1}^2}{\epsilon_1} \\ 0 & 0 & \frac{jm^2}{\epsilon_1} & j\frac{m\kappa}{\epsilon_1} \\ jm\kappa & -jk_{N1}^2 & 0 & 0 \\ -jm^2 & -jm\kappa & 0 & 0 \end{pmatrix} \qquad (2-145a)$$

$$\mathbf{A}_2 = \begin{pmatrix} 0 & 0 & -j\frac{m\kappa}{\epsilon_1}(2\Delta_1^0) & -j\frac{\kappa^2}{\epsilon_1}(2\Delta_1^0) \\ 0 & 0 & j\left[\frac{m^2}{\epsilon_1}(2\Delta_1^0) - 1\right] & j\frac{m\kappa}{\epsilon_1}(2\Delta_1^0) \\ 0 & j\epsilon_1(2\Delta_1^0) & 0 & 0 \\ j\epsilon_3 & 0 & 0 & 0 \end{pmatrix} \qquad (2-145b)$$

$$\mathbf{A}_4 = \begin{pmatrix} 0 & 0 & -j\frac{m\kappa}{\epsilon_1}(2\Delta_1^0)^2 & -j\frac{\kappa^2}{\epsilon_1}(2\Delta_1^0)^2 \\ 0 & 0 & \frac{jm^2}{\epsilon_1}(2\Delta_1^0)^2 & j\frac{m\kappa}{\epsilon_1}(2\Delta_1^0)^2 \\ 0 & 0 & 0 & 0 \\ -j\epsilon_3(2\Delta_3^0) & 0 & 0 & 0 \end{pmatrix} \qquad (2-145c)$$

$$\mathbf{A}_{2n} = \frac{(2\Delta_1^0)^n}{\epsilon_1}\begin{pmatrix} 0 & 0 & -jm\kappa & -j\kappa^2 \\ 0 & 0 & jm^2 & jm\kappa \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad n = 3, 4, 5, \ldots \qquad (2-145d)$$

and $\mathbf{A}_{2n-1} = 0$ for $n = 1, 2, 3, \ldots$ where $k_{N1}^2 = \epsilon_1 - \kappa^2$. Comparing eqs. (2-144) and (2-145) it can be readily seen that the series expansion for both cases are identical except for the presence of an additional element in $\mathbf{A}_2$ for the uniaxial graded-index case. The eigenvalues of A_0 for both cases are $\pm m$. Since the $\mathbf{A}_0$'s have repeated eigenvalues, in general, the choice for the matrix $\mathbf{P}_0$ should at best cause $\mathbf{B}_0$ to be a Jordan canonical matrix. Note, this is the only restriction which the solution method places on the form of $\mathbf{P}_0$. Any $\mathbf{P}_0$ which causes $\mathbf{B}_0$ to be a Jordan canonical matrix can be expected to result in a valid solution. Since it is posssible for several different choices of $\mathbf{P}_0$ to satisfy this condition, conceivably there may exist several possible mathematical solutions to the problem.

Since the solution for a step-index fiber exists as a special case of the solution for a graded-index fiber it is reasonable to choose $\mathbf{P}_0$ based on the knowledge of the exact solution for a step-index fiber. For the case of a uniaxial step-index fiber the exact solutions for E_z and h_z as given by eq. (2-30) suggest that $\mathbf{P}_0$ should be chosen so that the resulting $\mathbf{P}(s)$ yields $E_z = P_{11}v_1 + P_{13}v_3$ and $h_z = P_{32}v_2 + P_{34}v_4$ as solutions. If $\mathbf{P}_0$ is chosen as

$$\mathbf{P}_0 = \begin{pmatrix} k_{N1}^2 & 0 & k_{N1}^2 & 0 \\ m\kappa & jm & m\kappa & -jm \\ 0 & k_{N1}^2 & 0 & k_{N1}^2 \\ -jm\epsilon_1 & m\kappa & jm\epsilon_1 & m\kappa \end{pmatrix} \qquad (2-146)$$

this additional requirement is at the least satisfied for the lowest order solution where $\mathbf{P}(s) = \mathbf{P}_0$.

Using eq. (2-102) $\mathbf{B}_0$ is given by

$$\mathbf{B}_0 = \begin{pmatrix} m & 0 & 0 & 0 \\ 0 & m & 0 & 0 \\ 0 & 0 & -m & 0 \\ 0 & 0 & 0 & -m \end{pmatrix} . \qquad (2-147)$$

This is the more convenient form for $\mathbf{B}_0$ since eq. (2-88) can be easily satisfied by choosing $\mathbf{B}_n$ and $\mathbf{W}_n$ according to eq. (2-89) and hence $\mathbf{W}_n$ can be found using eqs. (2-92) and (2-93). If $\mathbf{B}_0$ was not a diagonal matrix then $\mathbf{W}_n$ would have to be found from a more complicated expression similiar to eq. (2-128).

Since $\mathbf{B}_0$ is a diagonal matrix, choosing $\mathbf{B}_n$ according to eq. (2-89a) makes $\mathbf{B}(s)$ a diagonal matrix. Therefore, in general, $\mathbf{B}(s)$ can be written as

$$\mathbf{B}(s) = \begin{pmatrix} B_{11}(s) & 0 & 0 & 0 \\ 0 & B_{22}(s) & 0 & 0 \\ 0 & 0 & B_{33}(s) & 0 \\ 0 & 0 & 0 & B_{44}(s) \end{pmatrix} \qquad (2-148a)$$

where

$$B_{ii}(s) = \pm m + \sum_{n=1}^{N} (\mathbf{B}_n)_{ii} s^n \qquad i = 1,2,3,4 , \qquad (2-148b)$$

N is the number of iterations and the upper(lower) sign corresponds to $i = 1,2 (i = 3,4)$. Using eq. (2-74) the differential equation for $\mathbf{v}(s)$ can be simply written as

$$\frac{dv_i}{ds} = \frac{1}{s} B_{ii}(s)\, ds \qquad i = 1,2,3,4 . \qquad (2-149)$$

The solution of eq. (2-149) for v_i is then given by

$$v_i(s) = C_i s^{\pm m} e^{\lambda_i(s)} \qquad i = 1,2,3,4 \qquad (2-150a)$$

where $C_i \quad i = 1,2,3,4$ is a constant,

$$\begin{aligned} \lambda_i(s) &= \int \frac{1}{s}[B_{ii}(s) \mp m]\, ds \qquad i = 1,2,3,4 \\ &= \sum_{n=1}^{N} (\mathbf{B}_n)_{ii} \frac{s^n}{n} \qquad i = 1,2,3,4 \qquad (2-150b) \end{aligned}$$

and the upper(lower) sign corresponds to $i = 1,2(i = 3,4)$. Since $\mathbf{u}(s)$ must be finite at $s = 0$ $\mathbf{v}(s)$ must also be finite at $s = 0$ and it is therefore necessary to set $C_3 = C_4 = 0$. The solution to eq. (2-65) can then be written as

$$\begin{pmatrix} E_z \\ sE_\phi \\ h_z \\ sh_\phi \end{pmatrix} = s^m \begin{pmatrix} P_{11}(s) & P_{12}(s) \\ P_{21}(s) & P_{22}(s) \\ P_{31}(s) & P_{32}(s) \\ P_{41}(s) & P_{42}(s) \end{pmatrix} \begin{pmatrix} C_1 e^{\lambda_1(s)} \\ C_2 e^{\lambda_2(s)} \end{pmatrix} \tag{2-151a}$$

where

$$P_{ij}(s) = \sum_{n=0}^{N} (\mathbf{P}_n)_{ij} s^n \qquad i = 1,2,3,4; \quad j = 1,2,3,4\,. \tag{2-151b}$$

For the example of a biaxial graded-index fiber the following expressions for $\lambda_i(s)$ and $P_{ij}(s)$ are obtained after two iterations

$$\lambda_1(s) = -\frac{1}{4m}\left[\frac{\epsilon_3}{\epsilon_1}(\epsilon_1 - \kappa^2) + \frac{m^2\kappa^2}{\epsilon_1 - \kappa^2}(2\Delta_1^0)\right] s^2 \tag{2-152a}$$

$$\lambda_2(s) = -\frac{1}{4m}\left[(\epsilon_1 - \kappa^2) - \frac{m^2\epsilon_1}{\epsilon_1 - \kappa^2}(2\Delta_1^0)\right] s^2 \tag{2-152b}$$

$$P_{11}(s) = (\epsilon_1 - \kappa^2) + \frac{1}{4m(m+1)}\left[\frac{\epsilon_3}{\epsilon_1}(\epsilon_1 - \kappa^2)^2 - m^2\kappa^2(2\Delta_1^0)\right] s^2 \tag{2-152c}$$

$$P_{12}(s) = -j\frac{m\kappa}{4}\left(\frac{m+2}{m+1}\right)(2\Delta_1^0)s^2 \tag{2-152d}$$

$$P_{21}(s) = m\kappa + \frac{\kappa}{4(m+1)}\left\{\frac{\epsilon_3}{\epsilon_1}(\epsilon_1 - \kappa^2) + \frac{m^2(2\Delta_1^0)}{\epsilon_1 - \kappa^2}\left[(\epsilon_1 - \kappa^2) + (m+1)\epsilon_1\right]\right\} s^2 \tag{2-152e}$$

$$P_{22}(s) = jm - \frac{j}{4(m+1)}\left\{(\epsilon_1 - \kappa^2) + \frac{m^2(2\Delta_1^0)}{\epsilon_1 - \kappa^2}\left[(m+1)\kappa^2 - (\epsilon_1 - \kappa^2)\right]\right\} s^2 \tag{2-152f}$$

$$P_{31}(s) = -j\frac{m^2\epsilon_1\kappa}{4(m+1)}(2\Delta_1^0)s^2 \tag{2-152g}$$

$$P_{32}(s) = (\epsilon_1 - \kappa^2) + \frac{1}{4m(m+1)}\left[(\epsilon_1 - \kappa^2)^2 - m^2\epsilon_1(2\Delta_1^0)\right] s^2 \tag{2-152h}$$

$$P_{41}(s) = -jm\epsilon_1 + \frac{j\epsilon_1}{4(m+1)}\left[\frac{\epsilon_3}{\epsilon_1}(\epsilon_1 - \kappa^2) - \frac{m^2(m+1)\kappa^2}{\epsilon_1 - \kappa^2}(2\Delta_1^0)\right]s^2 \qquad (2-152i)$$

$$P_{42}(s) = m\kappa + \frac{\kappa}{4(m+1)}\left[(\epsilon_1 - \kappa^2) - \frac{m^2(m+1)\epsilon_1}{\epsilon_1 - \kappa^2}(2\Delta_1^0)\right]s^2 . \qquad (2-152j)$$

This should not be considered an accurate solution for $\mathbf{u}(s)$ since the term Δ_3^0 does not appear anywhere in eq. (2-152). This solution is identical to the solution obtained after two iterations for a biaxial graded-index fiber where $\epsilon_1(s)$ has a parabolic profile and $\epsilon_2(s)$ and $\epsilon_3(s)$ are constant. Since Δ_3^0 only appears in the matrix $\mathbf{A}_4$ at least four iterations must be performed in order to obtain the effects of a non-constant ϵ_3. A solution for a uniaxial or a step index-fiber can be obtained from eq. (2-152) by setting Δ_1^0(and Δ_3^0) equal to zero.

For the example of a uniaxial graded-index fiber two iterations produce the following expression for $\lambda_i(s)$ and $P_{ij}(s)$

$$\lambda_1(s) = -\frac{1}{4m}\frac{\epsilon_3}{\epsilon_1}(\epsilon_1 - \kappa^2)s^2 \qquad (2-153a)$$

$$\lambda_2(s) = -\frac{1}{4m}(\epsilon_1 - \kappa^2)s^2 \qquad (2-153b)$$

and

$$P_{11}(s) = (\epsilon_1 - \kappa^2) + \frac{1}{4m(m+1)}\left[\frac{\epsilon_3}{\epsilon_1}(\epsilon_1 - \kappa^2)^2 - 2m^2\kappa^2(2\Delta_1^0)\right]s^2 \qquad (2-153c)$$

$$P_{12}(s) = -j\frac{m\kappa}{2(m+1)}(2\Delta_1^0)s^2 \qquad (2-153d)$$

$$P_{21}(s) = m\kappa + \frac{\kappa}{4(m+1)}\left[\frac{\epsilon_3}{\epsilon_1}(\epsilon_1 - \kappa^2) + 2m^2(2\Delta_1^0)\right]s^2 \qquad (2-153e)$$

$$P_{22}(s) = jm - \frac{j}{4(m+1)}\left[(\epsilon_1 - \kappa^2) - 2m^2(2\Delta_1^0)\right]s^2 \qquad (2-153f)$$

$$P_{31}(s) = j\frac{m\epsilon_1\kappa}{2(m+1)}(2\Delta_1^0)s^2 \qquad (2-153g)$$

$$P_{32}(s) = (\epsilon_1 - \kappa^2) + \frac{1}{4m(m+1)}(\epsilon_1 - \kappa^2)^2 s^2 \qquad (2-153h)$$

$$P_{41}(s) = -jm\epsilon_1 + \frac{j\epsilon_3}{4(m+1)}(\epsilon_1 - \kappa^2)s^2 \qquad (2-153i)$$

$$P_{42}(s) = m\kappa + \frac{\kappa}{4(m+1)}(\epsilon_1 - \kappa^2)s^2 \qquad (2-153j)$$

As was the case for the example of a biaxial fiber, the expressions in eq. (2-153) are not an accurate solution since the term Δ_3^0 does not appear in any equation. Again, in order to see the effects of $\epsilon_3(s)$ it is necessary to perform at least four iterations.

For the solutions of eq. (2-65) the generalized dispersion relation, given by eq. (2-25) can not be used. For the hybrid modes it is possible to derive from eq. (2-64) expressions for e'/e and h'/h' similar to eqs. (2-142) and (2-142). However, in general e'/e and h'/h are functions of the unknown constants C_1 and C_2 which appear in the general solution for $\mathbf{u}(s)$. For special cases, such as a step-index fiber, where E_z and h_z are decoupled it may be possible to set either C_1 or C_2 equal to zero without losing a a complete solution. The generalized dispersion relation can only be used if either C_1 or C_2 can be set to zero. Instead, using the solutions for eq. (2-65) a new dispersion relation must be derived by enforcing the electromagnetic boundary conditions at the core-cladding interface.

Using eq. (2-21), (2-22) and (2-151) the boundary conditions at $s = k_0a$ is satisfied provided

$$\begin{pmatrix} P_{11} & P_{12} & -K_m & 0 \\ P_{21} & P_{22} & \frac{m\kappa}{\gamma_N^2}K_m & j\frac{k_0a}{\gamma_N}K'_m \\ P_{31} & P_{32} & 0 & -K_m \\ P_{41} & P_{42} & -j\frac{\epsilon_c k_0 a}{\gamma_N}K'_m & \frac{m\kappa}{\gamma_N^2}K_m \end{pmatrix} \begin{pmatrix} C_1(k_0a)^m e^{\lambda_1} \\ C_2(k_0a)^m e^{\lambda_2} \\ C \\ D \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \qquad (2-154)$$

where $P_{ij} = P_{ij}(k_0a)$, $\lambda_i = \lambda_i(k_0a)$, $K_m = K_m(k_0a\gamma_N)$, $K'_m = K'_m(k_0a\gamma_N)$ and $\gamma_N^2 = \kappa^2 - \epsilon_c$. A non-trivial solution of eq. (2-154) exists whenever the determinant is equal to zero. An explicit equation for the determinant is not provided since it cannot be expressed

in a convenient form as was the case for the generalized dispersion relation derived in section 2.2.2. Instead the zeroes of the determinant are found directly from eq. (2-154).

2.3.6 Numerical Results

As was previously stated the solutions for the biaxial graded-index fiber and the uniaxial graded-index fibers given by eq. (2-152) and (2-153) do not include the effects of a non-constant $\epsilon_3(s)$. Obtaining a more accurate solution requires performing more than four iterations. For the example problem shown in section 2.3.3 performing more than four iterations is feasible since $\mathbf{A}_n = 0$ when $n > 2$ and $\mathbf{A}(s)$ is a 2×2 matrix. In contrast, performing more than two iterations in order to solve eq. (2-65) is much more difficult since $\mathbf{A}(s)$ is a 4×4 matrix and in general $\mathbf{A}_n \neq 0$ for $n > 2$ when $\epsilon_1(s)$ is not constant.

Instead of deriving algebraic equations for the elements of $\mathbf{F}_n$, $\mathbf{B}_n$, $\mathbf{W}_n$ and $\mathbf{P}_n$ the values of these matrices can be determined numerically if the values of m, κ and k_0a are known in advance. There are two difficulties with this appoach. First, numerical errors can develop since the accuracy of the matrices obtained in the i'th iteration depends upon the accuracy of the matrices obtained in the previous $i-1$ iterations. The second and more important problem comes from method in which the matrices $\mathbf{B}_n$ and $\mathbf{W}_n$ are chosen. As previously mentioned, it may not be possible to find $\mathbf{W}_n$ whenever $\lambda_i - \lambda_j - n = 0$ where λ_i and λ_j are eigenvalues of $\mathbf{A}_0$. In the analytic solution one could simply ignore the problem during the iteration process and then at the end of the process throw out the unbounded solutions with an appropriate choice of constants. The ability to do this appears to depend upon the form of $\mathbf{A}(s)$ and the ordering of the eigenvalues of $\mathbf{A}_0$ in $\mathbf{B}_0$. Luckily, due to the form of $\mathbf{A}(s)$ setting the third and fourth columns of $\mathbf{W}_n$ equal to zero is equivalent to

setting C_3 and C_4 equal to zero as was done in the solution of $\mathbf{v}(s)$ given by eq. (2-150).

Figures 11, 12 and 13 are plots of κ versus k_0a when $m = 1$ for the examples of the five possible types of fiber cores previously discussed. In each example $\epsilon_1 = n_1^2$ and $\epsilon_c = n_c^2$ where $n_1 = 1.515$ and $n_c = 1.5$. For the uniaxial fibers, step-index and graded-index, $\epsilon_3 = n_3^2$ where $n_3 = 2.0$. For the isotropic and uniaxial graded-index fibers all the permittivity profiles are parabolic. For the biaxial graded-index fiber ϵ_1 and ϵ_3 are parabolic while ϵ_2 is a constant. Due to the form of the solutions when $m \neq 0$ only the mode with the lowest cutoff frequency can be found for a given value of m, which for $m = 1$ is the HE_{11} mode. Based upon the extremely poor agreement between the asymptotic and exact solution method for a step-index fiber no results are given for the solutions of eqs. (2-66) and (2-67). For the case when $m = 0$ the asymptotic solution for a step index fiber is equivalent to finding a series solution for $\mathbf{u}(s)$. The poor agreement can then be attributed to using too few terms in the series to approximate the solution and can also be due to numerical errors from the evalution of the series.

In both figures 11 and 12 the asymptotic solutions for the isotropic and uniaxial step-index fibers are in good agreement with exact results. For the isotropic step-index fiber the HE_{11} mode for the asymptotic and exact solutions are almost identical when $k_0a < 10$. For $k_0a > 10$ the asymptotic solution begins to diverge from the exact solution but for $k_0a > 20$ the distance between the two curves is approximately constant. For the uniaxial step-index fiber the asymptotic solution begins to diverge from the exact solution around $k_0a = 8$ but the separation distance is reasonably constant for $k_0a > 20$.

Figure 13 is a plot of κ versus k_0a for an isotropic, a uniaxial and a biaxial graded-index

fiber. As was the case for the step-index fiber the HE_{11} mode for the uniaxial graded-index fiber is slightly displaced from the HE_{11} mode for the isotropic case. However, the displacement for the uniaxial graded-index fiber is not as large as the displacement for the uniaxial step-index fiber. A comparison of figure 13 with either figures 3 and 5 or figures 11 and 12 shows that for a given value of k_0a the value of κ for the HE_{11} mode in either an isotropic or uniaxial graded-index fiber is less than the value for the corresponding step-index fiber. Also for the HE_{11} mode κ as a function of k_0a for an isotropic or uniaxial graded-index fiber increases less rapidly than in a step-index fiber. This indicates the pulse delay which is proportional to $d\kappa/d(k_0a)$ is smaller for the HE_{11} mode in either type of graded-index fiber than in a step-index fiber.

For the biaxial graded-index fiber the HE_{11} mode is noticeably displaced from the HE_{11} modes for the isotropic and uniaxial graded-index fibers. A comparison with figure 3 shows that it lies approximately half the distance between the curves for the isotropic step-index fiber and the isotropic graded-index fibers.

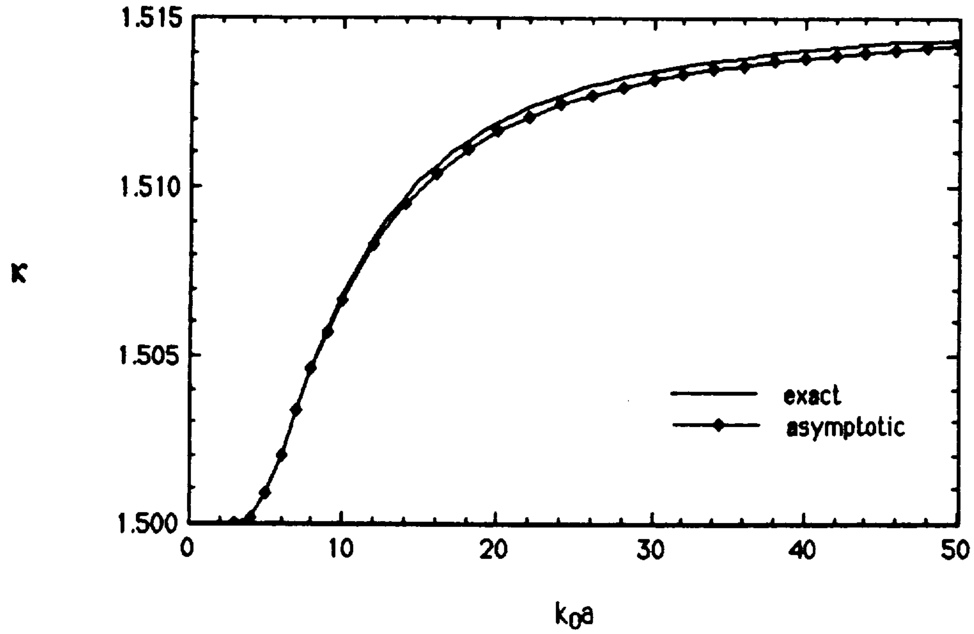

Figure 11 Asymptotic solution for isotropic step-index fiber: HE_{11} mode

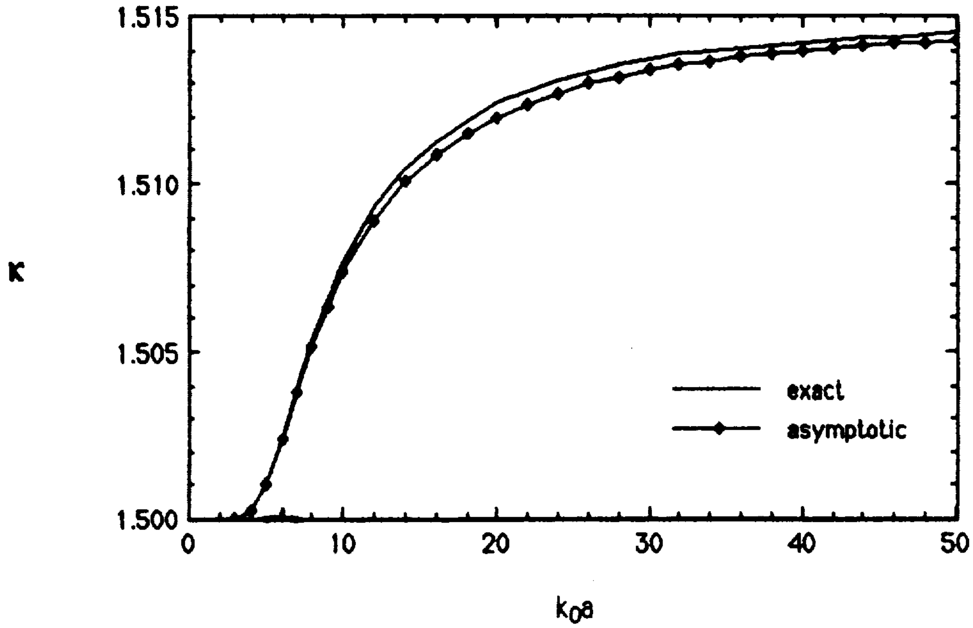

Figure 12 Asymptotic solution for uniaxial step-index fiber: HE_{11} mode

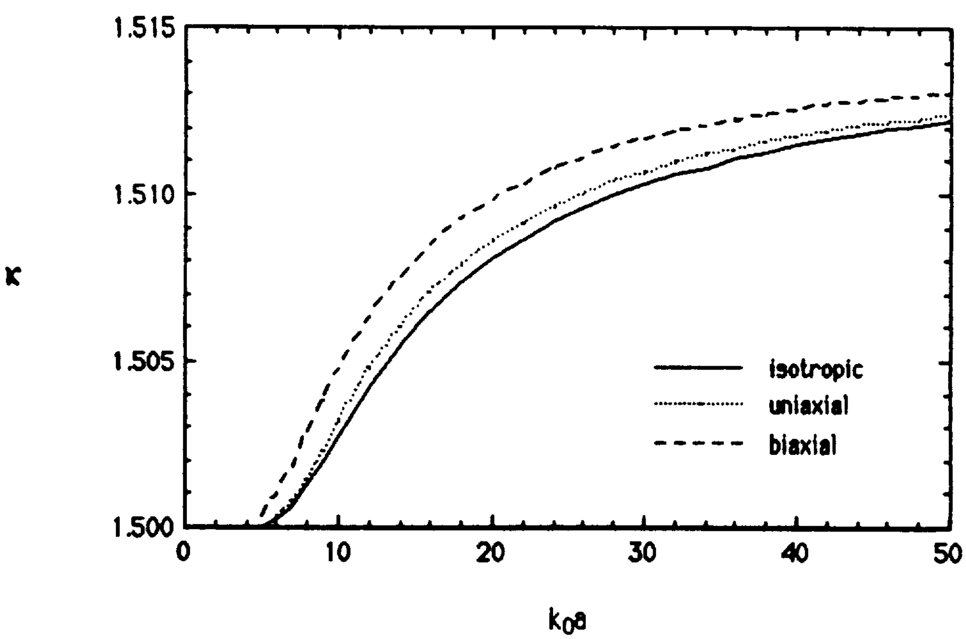

Figure 13 Asymptotic solutions for HE_{11} modes for isotropic, uniaxial and biaxial graded-index fibers

3. Stratification Technique

3.1 Formulation of Problem

A purely numerical approach to the problem of solving eq. (2-17) is to subdivide the core into N homogeneous layers as shown in figure 14 and then solve an easier problem in each layer [17]. Note, this solution method is valid for all modes of an isotropic or uniaxial fiber and the transverse modes, i.e. $m = 0$, for a biaxial graded-index-fiber. For the case $m \neq 0$ in a biaxial fiber eqs. (2-17a) and (2-17b) remain coupled and this solution method does not work.

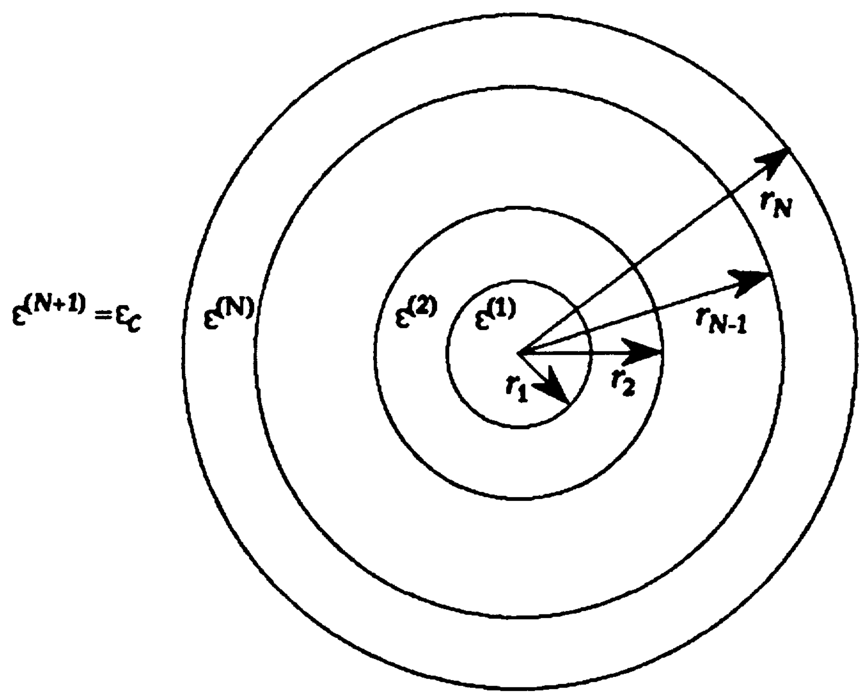

Figure 14 Geometry for stratification technique

For the case of an isotropic or uniaxial fiber $\epsilon_1(r) = \epsilon_2(r)$. In the n'th layer eq. (2-17)

can be written as

$$\frac{d^2E_z^{(n)}}{dr^2}+\frac{1}{r}\frac{dE_z^{(n)}}{dr}+\left\{\Lambda^2\left[\epsilon_3^{(n)}-\frac{\epsilon_3^{(n)}}{\epsilon_1^{(n)}}\kappa^2\right]-\frac{m^2}{r^2}\right\}E_z^{(n)}=0 \qquad (3-1a)$$

$$\frac{d^2h_z^{(n)}}{dr^2}+\frac{1}{r}\frac{dh_z^{(n)}}{dr}+\left\{\Lambda^2[\epsilon_3^{(n)}-\kappa^2]-\frac{m^2}{r^2}\right\}h_z^{(n)}=0 \qquad (3-1b)$$

where $\epsilon_i^{(n)}$, $i=1,3$, is the approximate value of $\epsilon_i(r)$ in the n'th layer. If we let

$$\left(p_1^{(n)}\right)^2=\Lambda^2\left[\epsilon_3^{(n)}-\frac{\epsilon_3^{(n)}}{\epsilon_1^{(n)}}\kappa^2\right], \qquad (3-2a)$$

$$\left(p_2^{(n)}\right)^2=\Lambda^2\left[\epsilon_1^{(n)}-\kappa^2\right], \qquad (3-2b)$$

$$\left(q_1^{(n)}\right)^2=-\left(p_1^{(n)}\right)^2 \qquad (3-3a)$$

and

$$\left(q_2^{(n)}\right)^2=-\left(p_2^{(n)}\right)^2 \qquad (3-3b)$$

then the solution of eq. (3-1) in the n'th layer is given by

$$E_z^{(n)}=\begin{cases}A_nJ_m(p_1^{(n)}r)+C_nY_m(p_1^{(n)}r), & \left(p_1^{(n)}\right)^2>0\\ A_nI_m(q_1^{(n)}r)+C_nK_m(q_1^{(n)}r), & \left(p_1^{(n)}\right)^2<0\end{cases} \qquad (3-4a)$$

$$h_z^{(n)}=\begin{cases}B_nJ_m(p_2^{(n)}r)+D_nY_m(p_2^{(n)}r), & \left(p_2^{(n)}\right)^2>0\\ B_nI_m(q_2^{(n)}r)+D_nK_m(q_2^{(n)}r), & \left(p_2^{(n)}\right)^2<0.\end{cases} \qquad (3-4b)$$

In order for the fields to be finite in the first layer C_1 and D_1 must be identically zero. In the cladding, $N+1$ layer, we require that $A_{N+1}=B_{N+1}=0$ and

$$\left(p_1^{(N+1)}\right)^2=\left(p_2^{(N+1)}\right)^2=\Lambda^2(\epsilon_c-\kappa^2)<0 \qquad (3-5)$$

so that the fields are exponentially decaying.

Recall that

$$\left(p_1^{(n)}\right)^2=\Lambda^2\left[\epsilon_3^{(n)}-\frac{\epsilon_3^{(n)}}{\epsilon_1^{(n)}}\kappa^2\right]=\Lambda^2\frac{\epsilon_3^{(n)}}{\epsilon_1^{(n)}}\left[\epsilon_1^{(n)}-\kappa^2\right].$$

But

$$\left(p_2^{(n)}\right)^2 = \Lambda^2\left[\epsilon_1^{(n)} - \kappa^2\right].$$

Therefore

$$\left(p_1^{(n)}\right)^2 = \frac{\epsilon_3^{(n)}}{\epsilon_1^{(n)}}\left(p_2^{(n)}\right)^2 = f^{(n)}\left(p_2^{(n)}\right)^2 \qquad (3-6)$$

where $f^{(n)} = \epsilon_3^{(n)}/\epsilon_1^{(n)}$. Since $\epsilon_1^{(n)}$ and $\epsilon_3^{(n)}$ are both positive in the core of the fiber $(p_1^{(n)})^2$ and $(p_2^{(n)})^2$ will always have the same sign. For $(p_2^{(n)})^2 > 0$ the tangential components of the electric and magnetic fields are given by

$$E_z^{(n)} = A_n J_m(\sqrt{f^{(n)}}p_2^{(n)}r) + C_n Y_m(\sqrt{f^{(n)}}p_2^{(n)}r) \qquad (3-7a)$$

$$h_z^{(n)} = B_n J_m(p_2^{(n)}r) + D_n Y_m(p_2^{(n)}r) \qquad (3-7b)$$

$$\begin{aligned} rE_\phi^{(n)} = {} & \frac{m\beta}{a(k_t^{(n)})^2}\left[A_n J_m(\sqrt{f^{(n)}}p_2^{(n)}r) + C_n Y_m(\sqrt{f^{(n)}}p_2^{(n)}r)\right] \\ & + \frac{jk_0 r p_2^{(n)}}{a(k_t^{(n)})^2}\left[B_n J'_m(p_2^{(n)}r) + D_n Y'_m(p_2^{(n)}r)\right] \qquad (3-7c) \\ rh_\phi^{(n)} = {} & \frac{m\beta}{a(k_t^{(n)})^2}\left[B_n J_m(p_2^{(n)}r) + D_n Y_m(p_2^{(n)}r)\right] \\ & - \frac{jk_0 r\epsilon_1^{(n)}\sqrt{f^{(n)}}p_2^{(n)}}{a(k_t^{(n)})^2}\left[A_n J'_m(\sqrt{f^{(n)}}p_2^{(n)}r) + C_n Y'_m(\sqrt{f^{(n)}}p_2^{(n)}r)\right] \qquad (3-7d) \end{aligned}$$

where $E_z^{(n)}$ and $h_z^{(n)}$ follow from eq. (3-1), $rE_\phi^{(n)}$ and $rh_\phi^{(n)}$ follow from eq. (2-8) and $(k_t^{(n)})^2 = k_0^2(\epsilon_1^{(n)} - \kappa^2)$. Note that eqs. (3-7c) and (3-7d) can be simplified by using the following

$$\epsilon_1^{(n)}\sqrt{f^{(n)}} = \epsilon_1^{(n)}\sqrt{\frac{\epsilon_3^{(n)}}{\epsilon_1^{(n)}}} = \sqrt{\epsilon_1^{(n)}\epsilon_3^{(n)}} \qquad (3-8)$$

and

$$\frac{p_2^{(n)}}{a(k_t^{(n)})^2} = \frac{ak_t^{(n)}}{a(k_t^{(n)})^2} = \frac{1}{k_t^{(n)}}. \qquad (3-9)$$

For $(p_2^{(n)})^2 < 0$ the tangential fields are given by eq. (3-7) provided we make the following substitutions

$$
\begin{array}{cc}
\multicolumn{2}{c}{p_2^{(n)} \rightarrow q_2^{(n)}} \\
J_m \rightarrow I_m, & J'_m \rightarrow I'_m \\
Y_m \rightarrow K_m, & Y'_m \rightarrow K'_m \\
\frac{m\beta}{a(k_t^{(n)})^2} \rightarrow -\frac{m\beta}{a(\gamma^{(n)})^2}, & \frac{jk_0 r}{k_t^{(n)}} \rightarrow -\frac{jk_0 r}{\gamma^{(n)}}
\end{array}
\qquad (3-10)
$$

where $(\gamma^{(n)})^2 = -(k_t^{(n)})^2$. Eq. (3-7) can be written in matrix form as

$$
\begin{pmatrix} E_z^{(n)} \\ h_z^{(n)} \\ rE_\phi^{(n)} \\ rh_\phi^{(n)} \end{pmatrix} = \mathbf{M}_n \begin{pmatrix} A_n \\ B_n \\ C_n \\ D_n \end{pmatrix}
$$

where $\mathbf{M}_n$ is the chain matrix for the n'th layer and is given by

$$
\mathbf{M}_n = \begin{pmatrix} c_1(r) & 0 & d_1(r) & 0 \\ 0 & e_1(r) & 0 & f_1(r) \\ k_1 c_1(r) & k_2 e_2(r) & k_1 d_1(r) & k_2 f_2(r) \\ -k_2\sqrt{\epsilon_1^{(n)}\epsilon_3^{(n)}}c_2(r) & k_1 e_1(r) & -k_2\sqrt{\epsilon_1^{(n)}\epsilon_3^{(n)}}d_2(r) & k_1 f_1(r) \end{pmatrix} \qquad (3-12)
$$

and

$$
\begin{array}{cc}
\text{for } (p_2^{(n)})^2 > 0 & \text{for } (p_2^{(n)})^2 < 0 \\
c_1(r) = J_m(\sqrt{f^{(n)}}p_2^{(n)}r) & c_1(r) = I_m(\sqrt{f^{(n)}}q_2^{(n)}r) \\
d_1(r) = Y_m(\sqrt{f^{(n)}}p_2^{(n)}r) & d_1(r) = K_m(\sqrt{f^{(n)}}q_2^{(n)}r) \\
e_1(r) = J_m(p_2^{(n)}r) & e_1(r) = I_m(q_2^{(n)}r) \\
f_1(r) = Y_m(p_2^{(n)}r) & f_1(r) = K_m(q_2^{(n)}r) \\
c_2(r) = J'_m(\sqrt{f^{(n)}}p_2^{(n)}r) & c_2(r) = I'_m(\sqrt{f^{(n)}}q_2^{(n)}r) \; . \\
d_2(r) = Y'_m(\sqrt{f^{(n)}}p_2^{(n)}r) & d_2(r) = K'_m(\sqrt{f^{(n)}}q_2^{(n)}r) \\
e_2(r) = J'_m(p_2^{(n)}r) & e_2(r) = I'_m(q_2^{(n)}r) \\
f_2(r) = Y'_m(p_2^{(n)}r) & f_2(r) = K'_m(q_2^{(n)}r) \\
k_1 = \frac{m\beta}{a(k_t^{(n)})^2} & k_1 = -\frac{m\beta}{a(\gamma^{(n)})^2} \\
k_2 = \frac{jk_0 r}{k_t^{(n)}} & k_2 = -\frac{jk_0 r}{\gamma^{(n)}}
\end{array}
\qquad (3-13)
$$

Matching the tangential fields at the surfaces of each layer gives

$$
\begin{aligned}
\mathbf{M}_1(r_1)\begin{pmatrix} A_1 \\ B_1 \\ 0 \\ 0 \end{pmatrix} &= \mathbf{M}_2(r_1)\begin{pmatrix} A_2 \\ B_2 \\ C_2 \\ D_2 \end{pmatrix} \\
\mathbf{M}_2(r_2)\begin{pmatrix} A_2 \\ B_2 \\ C_2 \\ D_2 \end{pmatrix} &= \mathbf{M}_3(r_2)\begin{pmatrix} A_3 \\ B_3 \\ C_3 \\ D_3 \end{pmatrix} \\
&\vdots \\
\mathbf{M}_N(r_N)\begin{pmatrix} A_N \\ B_N \\ C_N \\ D_N \end{pmatrix} &= \mathbf{M}_{N+1}(r_N)\begin{pmatrix} 0 \\ 0 \\ C_{N+1} \\ D_{N+1} \end{pmatrix}
\end{aligned}
\qquad (3-14)
$$

where r_n is the normalized radius of the n'th layer. Eq. (3-14) is essentially a system of $4N$ equations in $4N$ unknowns. The propagating modes can be found directly from eq. (3-14) by setting the determinant of the system matrix equal to zero. The time required to find a determinant of a $n \times n$ matrix is proportional to n^3. If we double the number of layers then it takes 8 times as much time to find the determinant. What is needed is a more efficient algorithm for determining the allowable modes from eq. (3-14).

If we recognize that the i'th coefficient vector can be written in terms of the $i+1$'th coefficient vector as

$$
\begin{pmatrix} A_i \\ B_i \\ C_i \\ D_i \end{pmatrix} = \mathbf{M}_i^{-1}(r_i)\mathbf{M}_{i+1}(r_i)\begin{pmatrix} A_{i+1} \\ B_{i+1} \\ C_{i+1} \\ D_{i+1} \end{pmatrix} . \qquad (3-15)
$$

then eq. (3-14) can be more conveniently written as

$$
\begin{aligned}
\mathbf{M}_1(r_1)\begin{pmatrix} A_1 \\ B_1 \\ 0 \\ 0 \end{pmatrix} &= \mathbf{M}_2(r_1)\mathbf{M}_2^{-1}(r_2)\mathbf{M}_3(r_2)\mathbf{M}_3^{-1}(r_3)\cdots \\
&\cdots\mathbf{M}_N(r_{N-1})\mathbf{M}_N^{-1}(r_N)\mathbf{M}_{N+1}(r_N)\begin{pmatrix} 0 \\ 0 \\ C_{N+1} \\ D_{N+1} \end{pmatrix} . \qquad (3-16)
\end{aligned}
$$

Defining an overall chain matrix product, $\mathcal{M}$, where

$$\mathcal{M} = \mathbf{M}_2(r_1)\prod_{i=2}^{N}\mathbf{M}_i^{-1}(r_i)\mathbf{M}_{i+1}(r_i) \qquad (3-17)$$

eq. (3-16) can then be written as

$$\mathbf{M}_{bnd}\begin{pmatrix} A_1 \\ B_1 \\ C_{N+1} \\ D_{N+1}\end{pmatrix} =$$

$$= \begin{pmatrix} (\mathbf{M}_1)_{11} & (\mathbf{M}_1)_{12} & -(\mathcal{M})_{13} & -(\mathcal{M})_{14} \\ (\mathbf{M}_1)_{21} & (\mathbf{M}_1)_{22} & -(\mathcal{M})_{23} & -(\mathcal{M})_{24} \\ (\mathbf{M}_1)_{31} & (\mathbf{M}_1)_{32} & -(\mathcal{M})_{33} & -(\mathcal{M})_{34} \\ (\mathbf{M}_1)_{41} & (\mathbf{M}_1)_{42} & -(\mathcal{M})_{43} & -(\mathcal{M})_{44} \end{pmatrix}\begin{pmatrix} A_1 \\ B_1 \\ C_{N+1} \\ D_{N+1}\end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0\end{pmatrix} \qquad (3-18)$$

where $(\mathbf{M}_1)_{ij}$ is an element of $\mathbf{M}_1$ and $(\mathcal{M})_{ij}$ is an element of $\mathcal{M}$. The problem has been effectively reduced from a system of $4N$ equations in $4N$ unknowns into a system of 4 equations in 4 unknowns. Propagating modes are found by requiring the determinant of $\mathbf{M}_{bnd}$ to be identically equal to zero. For the special case of an isotropic or uniaxial step-index fiber $N = 1$ and eq. (3-18) reduces to either eq. (2-29) for the isotropic case or eq. (2-31) for the uniaxial case.

From eq. (3-17) it can be seen that calculating the overall chain matrix, $\mathcal{M}$, requires $N-1$ matrix inversions and $2(N-1)$ matrix multiplications. Since the size of individual matrices is fixed the time needed to find $\mathcal{M}$ and hence find the determinant of $\mathbf{M}_{bnd}$ grows linearly with increasing N. The chain matrix approach is therefore a more desirable algorithm for determining what modes propagate.

3.2 Numerical Results

The accuracy of this solution method increases with the number of layers, however, with this increase in accuracy comes an increase in computation time. For a parabolic profile, a choice of five layers gives reasonable accuracy with a minimum of computation time [22]

Figures 15 and 16 are plots of κ versus k_0a for the cases $m = 0$ and 1 of an isotropic graded-index fiber with a parabolic permittivity profile. As in the previous example the values of n_r and n_c are taken to be 1.515 and 1.5 respectively. Comparing figures 15 and 16 with figures 2 and 3 several differences can be seen in the dispersion curves for the step-index and graded-index fibers. The most important difference is the value of κ as a function of k_0a increases less rapidly for the graded-index fiber than for the step-index fiber. This indicates that the pulse delay which is proportional to $d\kappa/d(k_0a)$ is smaller for an isotropic fiber with a parabolic permittivity profile than one with a step profile. The other notable features are the increase in the cutoff frequencies of all modes except the HE_{11} compared with the step-index fiber and several of the hybrid modes have become degenerate or nearly degenerate.

Figures 17 and 18 are plots of κ versus k_0a for a uniaxial graded-index fiber with parabolic permittivity profiles where $n_1 = 1.515$, $n_3 = 2.0$ and $n_c = 1.5$. Comparing figures 15 and 16 with figures 3 and 4 it can be seen that like the isotropic graded-index fiber the value of κ versus k_0a increases less rapidly than in a uniaxial step-index fiber. The uniaxial graded-index also exhibits an increase in the cutoff frequencies for all modes except the HE_{11} mode as compared with the uniaxial step-index fiber.

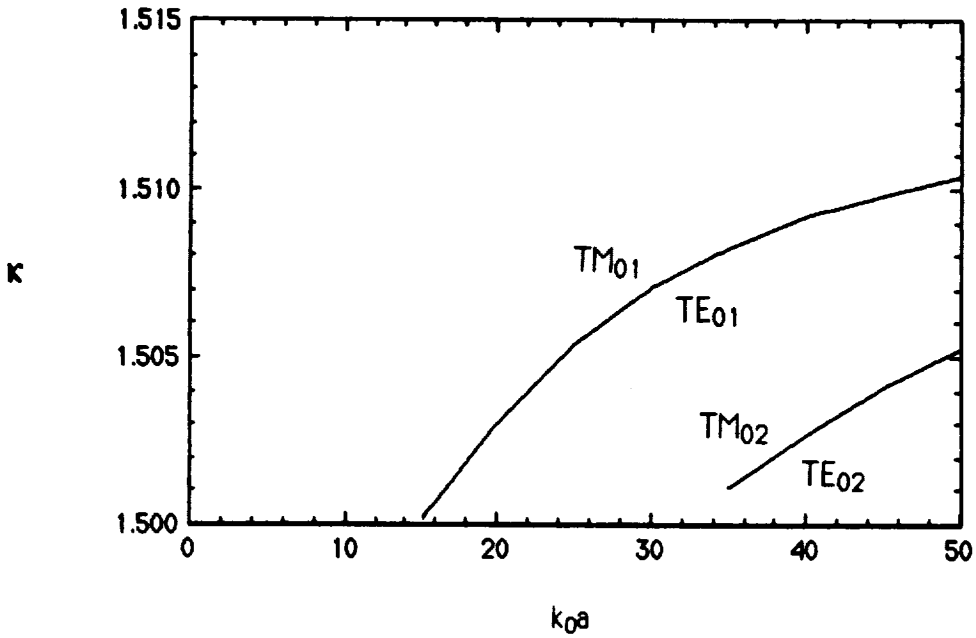

Figure 15 Stratification solution for isotropic parabolic-index fiber: $m = 0$

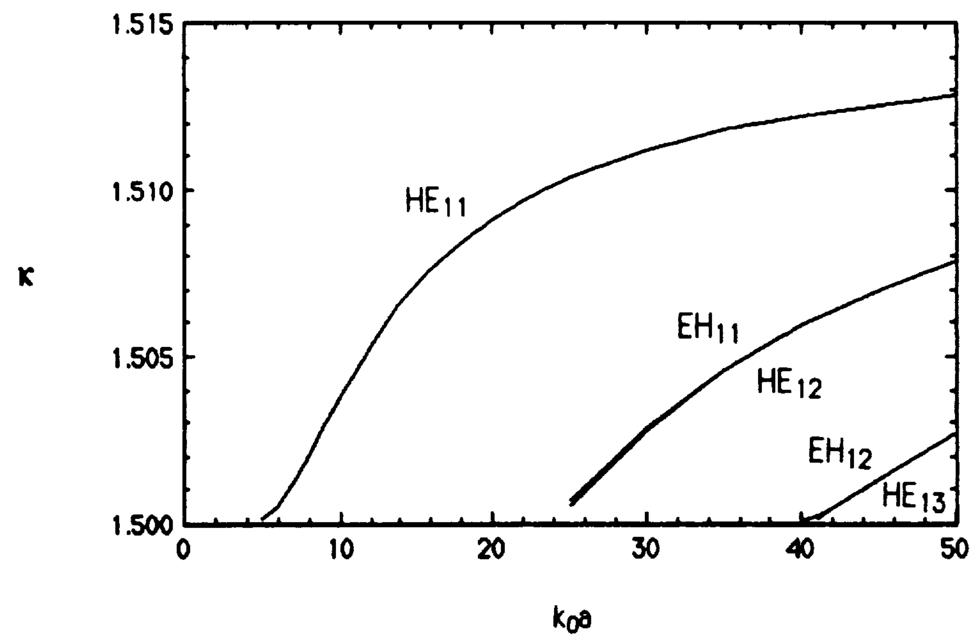

Figure 16 Stratification solution for isotropic parabolic-index fiber: $m = 1$

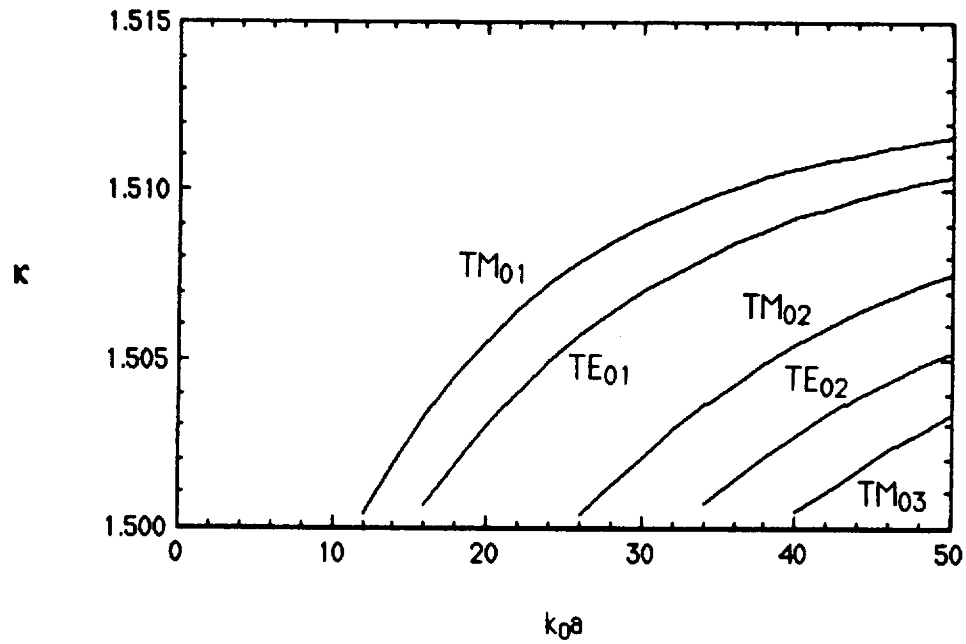

Figure 17 Stratification solution for uniaxial parabolic-index fiber: $m = 0$

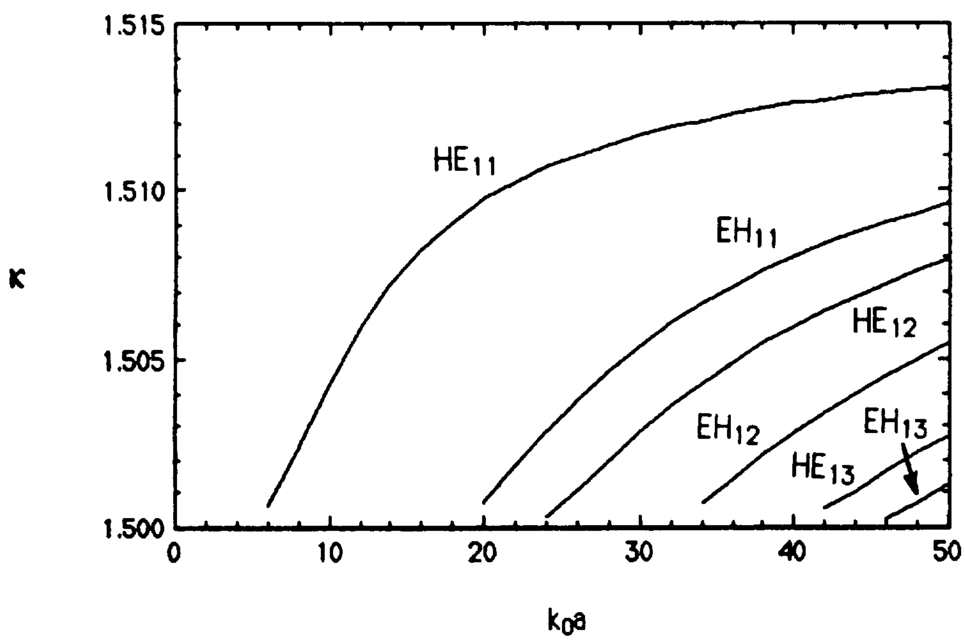

Figure 18 Stratification solution for uniaxial parabolic-index fiber: $m = 1$

4. Discussion

A total of five types of fiber cores have been discussed. For an isotropic or uniaxial step-index fiber either the wave equation formulation or the matrix equation formulation can be solved exactly in terms of Bessel functions. The wave equation formulation can be solved using WKB analysis for all types except a biaxial graded-index fiber. The matrix equation formulation can be solved for all five types of fibers using the method of asymptotic partitioning of systems of equations.

For an isotropic or uniaxial graded-index fiber the WKB solutions are solution of an associated scalar wave equation not the full vector problem as given in eqs. (2-17) and (2-18). For an isotropic graded-index fiber when the permittivity profile is parabolic and the fields are assumed to be far from cutoff an approximate solution of the vector problem is possible. A comparison of the WKB solutions and the vector solutions shows the vector solutions can be obtained from the WKB analysis by renumbering the WKB modes. Strictly speaking this comparison is valid only for an isotropic parabolic-index fiber. However, it seem reasonable to extend the comparison to isotropic graded-index fibers where the permittivity profiles are not parabolic and also to uniaxial graded-index fibers. This renumbering of the WKB modes has been used as the basis for a more general vector analysis of an isotropic graded-index fiber using a generalized WKB technique [10], [15].

The negative aspect to the WKB solutions lies with the assumption that the core is infinite in extent and therefore there is no need to impose boundary conditions on the electric and magnetic fields. For a WKB solution the allowable values of κ are determined in the process of determining the solution. In contrast, for a step-index fiber, where an

exact solution is possible, the allowable values of κ are determined by imposing boundary conditions. If the renumbered WKB modes are compared with results obtained using the stratification technique one finds a poor agreement between the two methods. For example according to eq. (2-61) an HE_{1n} mode is equivalent to a $\mathrm{WKB}_{0,n-1}$ mode. However, a comparison of figures 6 and 16 shows the HE_{12} and HE_{13} modes do not correspond to the WKB_{01} and WKB_{01} modes. The HE_{12} and HE_{13} modes do agree very well with the WKB_{02} and WKB_{04} respectively. This suggests the boundary conditions are important even when a mode is far from cutoff. This suggests that further investigation is needed to determine whether the WKB modes can be renumbered in such a way as to be valid for a graded-index fiber with a finite core and cladding.

In theory the matrix equation can be solved for any type of fiber core using the method of partitioning of systems of equations. The solution obtained is valid wherever the Taylor series expansion for the matrix $\mathbf{A}(s)$ is valid. Simply because it is theoretically possible to solve eq. (2-65) does not mean the solutions obtained are of practical interest. It would be useful to compare the asymptotic solution of eq. (2-65) with some known solution, preferably an exact solution, in order to determine whether a valid solution has been found. The only case where an exact solution is known is for a step-index fiber.

The asymptotic solution for a step-index fiber can be obtained as a special case of the asymptotic solution for either the uniaxial parabolic-index fiber, eq. (2-152) or the biaxial graded-index fiber, eq. (2-153). Setting Δ_1^0 and Δ_3^0 equal to zero in either eq. (2-152) or (2-153) the asymptotic solutions for E_z and h_z for a step-index fiber can be written as

$$E_z = C_1 s^m P_{11}(s) e^{\lambda_1(s)} \qquad (4-1a)$$

$$h_z = C_2 s^m P_{32}(s) e^{\lambda_1(s)} \qquad (4-1b)$$

where

$$\lambda_1(s) = -\frac{1}{4m}\frac{\epsilon_3}{\epsilon_1}(\epsilon_1 - \kappa^2)s^2 \qquad (4-1c)$$

$$\lambda_2(s) = -\frac{1}{4m}(\epsilon_1 - \kappa^2)s^2 \qquad (4-1d)$$

$$P_{11}(s) = (\epsilon_1 - \kappa^2)\Big[1 + \frac{1}{4m(m+1)}\frac{\epsilon_3}{\epsilon_1}(\epsilon_1 - \kappa^2)s^2\Big] \qquad (4-1e)$$

$$P_{32}(s) = (\epsilon_1 - \kappa^2)\Big[1 + \frac{1}{4m(m+1)}(\epsilon_1 - \kappa^2)s^2\Big] \qquad (4-1g)$$

and $m > 0$. Comparing eq. (4-1) with the asymptotic solution of Bessel's equation given by eq. (2-114) E_z and h_z can be written as

$$E_z = k_{N1}^2 C_1 y(pk_{N1}s) \qquad (4-2a)$$

$$h_z = k_{N1}^2 C_2 y(k_{N1}s) \qquad (4-2b)$$

where $k_{N1}^2 = \epsilon_1 - \kappa^2$, $p^2 = \epsilon_3/\epsilon_1$ and $y(x)$ is the asymptotic solution for the Bessel function J_m and is given by

$$y(x) = \frac{1}{2}x^m\left[1 + \frac{x^2}{4m(m+1)}\right]e^{-\frac{x^2}{4m}}. \qquad (4-2c)$$

Since the exact solution for E_z and h_z are proportional to $J_m(pk_{N1}s)$ and $J_m(k_{N1}a)$ respectively the asymptotic solutions for E_z and h_z appear correct.

In general, when $m \neq 0$ the solutions to the matrix equation using the method of asymptotic partioning of systems of equations appear to be of the general form

$$y(x) = p(x)e^{-q(x)} \qquad (4-3)$$

where $p(x)$ is a monotonic function and $q(x)$ is a positive real monotonically increasing function. This implies the solutions will not behave in an oscillatory manner. For example,

the asymptotic solution to Bessel's equation accurately reproduces the form of $J_m(x)$ in the region $0 < x < x_1 - \delta$ where x_1 is the first zero of J_m and δ is a small positive number which depends on the order of the solution. In the region $x > x_1 - \delta$ the asymptotic solution falls rapidly to zero and can be taken to be identically equal to zero a short distance past x_1. As a consequence only the mode with the lowest cutoff frequency for a given value of m will be found when the boundary conditions are imposed.

This is main disadvantage in solving the matrix equation using the method of asymptotic partitioning of systems of equations. However, the solutions obtained are in good agreement with exact and numerical results. On the other hand, the WKB analysis of the wave equation produces results which can not be directly compared with either exact or numerical results.

A potental area of further research involves the comparison of the asymptotic solutions of the matrix equation with the asymptotic forms of well known functions in an attempt either to simplify the problem so as to get better asymptotic solutions or to determine the form of the exact solution.

Cited Literature

1. Snitzer, E.: Cylindrical dielectric waveguide modes. *J. Opt. Soc. Amer.*, 51, no. 5: 491-498, 1961

2. Paul, D.K., and Shevgaonkar, R.K.: Multimode propagation in anisotropic optical waveguides. *Radio Sci.*, 16: 525-533, 1981.

3. Tien, P.K., Gordon, J.P. and Whinnery, J.R.: Focusing of a light beam of gaussing field distribution in continuous and parabolic lens-like media. *Proc. IEEE*, 53: 129-136, 1965

4. Streifer, W and Kurtz, C.N.: Scalar analysis of radially inhomogeneous guiding media. *J. Opt. Soc. Am.*, 57, no :779-789, 1967

5. Cherin, A.H.: *An Introduction to Optical Fibers.* McGraw Hill, 1983

6. Yariv, A.: *Optical Electronics.* Holt, Reinhart and Winston, Inc., 1985

7. Kurtz, C. and Streifer, W.: Guided waves in inhomogeneous focusing media Part I: Formulation, solution for quadratic inhomogeneity. *IEEE Trans. Microwave Theory Tech.*, MTT-17, no. 1: 11-15, 1969

8. Kurtz, C. and Streifer, W.: Guided Waves in inhomogeneous focusing media Part II: Asymptotic solution for general weak inhomogeneity. *IEEE Trans. Microwave Theory Tech.*, MTT-17, no. 5: 250-253, 1969

9. Yip, G.L. and Nemoto, S.: The relations between scalar modes in a lenslike medium and vector modes in a self-focusing optical fiber. *IEEE Trans. Microwave Theory Tech.*, MTT-23, no. 2: 260-263, 1975

10 Hashimoto, M.: Asymptotic vector modes in inhomogeneous circular waveguides. *Radio Sci.*, 17, no. 1: 3-9, 1982

11. Hashimoto, M.: Hybrid modes of graded-index optical fibres. *Electron. Lett.*, 17, no. 18: 659-660, 1981

12. Hashimoto, M.: Vector wave characteristics of radially inhomogeneous waveguide modes. *Electron. Lett.*, 16, no. 13: 494-495, 1980

13. Ikuno, H.: Asymptotic eigenvalues of vector wave equation for guided modes in graded-index fibre. *Electron. Lett.*, 17, no. 1: 8-9, 1981

14. Ikuno, H.: Propagation constants of guided modes in graded-index fibre. *Electron. Lett.*,

15, no. 23: 762-763, 1979

15 Ikuno, H.: Vectorial wave analysis of graded-index fibers. *Radio Sci.*, 17, no. 1: 37-42, 1982

16. Tonning, A.: Circularly symmetric optical waveguide with strong anisotropy. *IEEE Trans. Microwave Theory Tech.*, MTT-30, no. 5: 790-794, 1982

17. Yeh, C. and Lindgren, G.: Computing the propagation characteristic of radially stratified fibers: an efficient method. *Applied Optics*, 16, no. 2: 483-493, 1977

18. Tonning, A.: An altenative theory of optical waveguides with radial inhomogeneities. *IEEE Trans. Microwave Theory Tech.*, MTT-30, no. 5: 781-789, 1982

19. Safaai-Jazi, A. and Yip, G.L.: Classification of hybrid modes in cylindrical dielectric optical waveguides. *Radio Sci.*, 12, no 4: 603-609, 1977

20. Mathews, J. and Walker, R.L.: *Mathematical Methods of Physics.* Benjamin/Cummings, 1964

21. Kawakami, S. and Nishizawa, J.: An optical waveguide with optimum distribution of the refractive index. *IEEE Trans. Microwave Theory Tech.*, MTT-16, no. 10:814-818, 1968

22. Nayfeh, A.: *Perturbation Methods.* John Wiley & Sons, Inc., 1973

Appendix: Computer Programs

This appendix contains three programs, `WKB`, `ASYMP`, and `STRAT` which were used to find the dispersion curves in figures 3-13 and 15-18. The program `WKB` numerically solves the mode condition, eq. (2-43) for either isotropic or uniaxial graded-index fibers where the permittivity profiles are given by eq. (2-44). The program `ASYMP` uses a numerical implementation of the method of asymptotic partitioning of systems of equations to solve eq. (2-65) and then finds the allowable modes using eq. (2-154). The type of fiber core for which `ASYMP` solves eq. (2-65) is determined by the procedure `findAn`. A version of `findAn` is given for the cases of a uniaxial step-index, a uniaxial graded-index fiber with parabolic permittivity profiles and a biaxial graded-index fiber where $\epsilon_2(s)$ is a constant and $\epsilon_1(s)$ and $\epsilon_3(s)$ have a parabolic profile. The program `STRAT` implements the chain matrix version of the stratification technique described in section 3. The dispersion curves for the step-index fibers, figures 2-5, are obtained using `STRAT` with the number of layers equal to one.

All three programs make use of some or all of the following IMSL subroutines:

`bsj0`	Bessel function of the first kind, order zero
`bsj1`	Bessel function of the first kind, order one
`bsjs`	Bessel function of the first kind, real order
`bsy0`	Bessel function of the second kind, order zero
`bsy1`	Bessel function of the second kind, order one
`bsys`	Bessel function of the second kind, real order
`bsi0`	Modified Bessel function of the first kind, order zero
`bsi1`	Modified Bessel function of the first kind, order one
`bsis`	Modified Bessel function of the first kind, real order
`bsk0`	Modified Bessel function of the second kind, order zero
`bsk1`	Modified Bessel function of the second kind, order one
`bsks`	Modified Bessel function of the second kind, real order
`lftcg`	Find LU factorization for a complex general matrix
`lfdcg`	Find determinant of a complex general matrix from LU factorization
`mcrcr`	Multiply two complex rectangular matrices
`ccgcg`	Copy complex general matrix
`zbren`	Find zero of a real function which changes sign over a given interval
`zreal`	Find zeroes of a real function
`qdag`	Integrate real function using adaptive quadrature

If any of the above subroutine names is preceeded by the letter `d`, such as `dbsjs`, then the double precision version is being used. In addition `ASYMP` has it's own procedures to perform addition, subtraction and multiplication for complex numbers and complex matrices and also a procedure to find the determinant of a complex matrix.

```
C***********************************************************************
C
C This program integerates the phase term of a WKB solution for
C a graded index fiber in order to find the dispersion curve
C
C***********************************************************************
C
C     Global Constants
C
      integer numMax, dSize, maxLvl, maxPnt, iso, uniaxl
      real pi
C
      parameter ( numMax=5, dSize=100 )
      parameter ( iso = 1, uniaxl = 2 )
C
C     Input Parameters
C
C       type = 1 for isotropic fiber
               2 for uniaxial fiber
C       n(1) = maximum value of the refractive index of the core
C              in the rho direction
C       n(2) = maximum value of the refractive index of the core in
C              the phi direction
C       nc = refractive index of the cladding
C       mu = mode order of the solution
C       alf(n) = parameters which describes the shape of the
C                refractive index profiles
C       Kamin = minimum value of Ka
C       Kamax = maximum value of Ka
C       numKa = number of divisions between Kamax and Kamin
C       KppMin = minimum value of normalized propagation constant
C       KppMax = maximum value of normalized propagation constant
C       numKpp = number of division between KppMin and KppMax
C
      real n(3), alf(3), nc, Kamax, Kamin, KppMax, KppMin
      integer mu, numKa, numKpp, type
C
C     Computed Parameters
C
C       e(i) = maximum value of permittivity in the core
C            = n(i)**2
C       ec = permittiviy of cladding = nc**2
C       delKa = increment for Ka = ( Kamax-Kamin )/numKa
C       delKpp = increment for kappa = ( KppMax-KppMin )/numKpp
C
      real e(3), ec, delKa, delKpp
C
C     Program Variables
C
C       Ka = ko * a = normalized wave number
C       kappa = normalized propagation constant = B/ko
C
C       i,j,k,loopKp,loopKa = loop variables
C
      real Ka, kappa, Kpp(dsize), IntDvPi(dSize,2), slope, Ksoln,
     &     result, delY
      integer i, loopKa, loopKp
      logical*1 flag(2), modes(2), iroots, uroots
C
C     Root finding and integration parameters
C
      integer irule
      parameter( irule=2 )
C
      real errabs, errrel, errest, r1(2), r2(2), eps, eta
      parameter ( errabs=0.001, errrel=0.001 ,eps=1.0e-07, eta=0.1 )
C
C
C     Phase function declarations
```

```
C
      real psi1, psi2
      external psi1, psi2
      common kappa, Ka, e, ec, alf, mu, type
C
C..............................................................
C
      pi = acos( -1.0 )
C
C..............................................................
C
C     Read input parameters
C
      read (53,*) type
      do 10 i = 1, 3
        read (53,*) n(i), alf(i)
        print 101, i, n(i), i, alf(i)
        e(i) = n(i)**2
        if ( alf(i) .lt. 1 ) then
            print 102, i
            stop
          endif
 10   continue
      if ( n(1) .ne. n(2) ) then
          print *, '>>Error: n(1) and n(2) must be equal'
          stop
        endif
C
      read (53,*) mu, nc
      print *, 'nc = ', nc
      print *, 'mu = ', mu
      ec = nc**2
      do 11 i = 1, 3
        if ( n(i) .le. nc ) then
            print 104, i
            stop
          endif
 11   continue
C
      read (53,*) KaMin, KaMax, numKa
      read (53,*) KppMin, KppMax, numKpp
      if ( numKpp .gt. dsize ) numKpp = dsize
C
      delKa = (KaMax-KaMin)/numKa
      delKpp = (KppMax-KppMin)/numKpp
C
C     Loop through values of Ka
C
      delKa = ( Kamax-Kamin )/numKa
      Ka = Kamin
      do 70 loopKa = 1, numKa+1
C
C       Loop through values of B
C
        kappa = KppMax + delKpp
        flag(1) = .false.
        flag(2) = .false.
C
        do 40 loopKp = 1, numKpp
          kappa = kappa - delKpp
          Kpp(loopKp) = kappa
          IntDvPi(loopKp,1) = 0.0
          IntDvPi(loopKp,2) = 0.0
C
C         Find turning points r  and r  for psi1 and psi2
C                              1      2
C
          modes(1) = .false.
          modes(2) = .false.
```

```
          if ( mu .eq. 0 ) then
              do 30 i = 1, 2
                if ( n(i) .ge. kappa ) then
                    r2(i) = ((e(i)-kappa**2)/(e(i)-ec))**(1.0/alf(i))
                    r1(i) = -1.0*r2(i)
                    modes(i) = .true.
                  else
                    r1(i) = 0.0
                    r2(i) = 0.0
                  endif
 30           continue
              if ( type .eq. iso ) then
                  modes(2) = .false.
                endif
            else
              if ( type .eq. iso ) then
                  modes(1) = iroots( r1(1), r2(1), 1 )
                else
                  modes(1) = uroots( r1(1), r2(1) )
                  modes(2) = iroots( r1(2), r2(2), 2 )
                endif
            endif
C
C         Integrate phase terms from r  to r
C                                     1     2
C
          if ( modes(1) ) then
                  call qdag( psi1, r1(1), r2(1), errabs, errrel, irule,
     &                       result, errest )
                endif
              IntDvPi(loopKp,1) = result/pi
              flag(i) = .true.
            else
              IntDvPi(loopKp,1) = 0.0
            endif
          if ( modes(2) ) then
                  call qdag( psi1, r1(2), r2(2), errabs, errrel, irule,
     &                       result, errest )
                endif
              IntDvPi(loopKp,2) = result/pi
              flag(i) = .true.
            else
              IntDvPi(loopKp,2) = 0.0
            endif
C
 40     continue
C
C       Determine values of kappa for which the integral between the
C       turning points satisfies the phase condition.
C
        do 60 i = 1, 2
          if ( flag(i) ) then
              delY = 0.5
 45           if ( ( intDvPi(1,i) - delY ) .gt. 0.0 ) then
                  delY = delY + 1.0
                  goto 45
                endif
              do 50 loopKp = 2, numKpp
                if ( ( intDvPi(loopKp,i) - delY ) .gt. 0.0 ) then
                    slope = (intDvPi(loopKp-1,i) - intDvPi(loopKp,i))
     &                        / ( Kpp(loopKp-1) - Kpp(loopKp) )
                    Ksoln = Kpp(loopKp)+(delY-intDvPi(loopKp,i))/slope
                    delY = delY + 1.0
                    print 100, i, Ka, Ksoln
                  endif
 50           continue
            endif
 60     continue
```

```
C
        Ka = Ka + delKa
 70   continue
C
C.............................................................................
C
 100  format( 1X, '>>Phase condition satisfied for psi', I1,
     &        ' at Ka = ', F6.2, ' B/Ko = ', F16.10 )
 101  format( 1X, 'n(', I1, ') = ', f8.6, ', alf(', I1, ') = ', F5.3 )
 102  format( 1X, '>>Error: alf(', I1, ') must be greater than 0' )
 103  format (1X, '>>Error: n(', I1, ') must be greater than nc' )
C
C.............................................................................
C
      stop
      end
C
C*****************************************************************************
C
      logical function iroots( r1, r2, select )
C
C*****************************************************************************
C
      real r1, r2
      integer select
C
      real delR, delta, errabs, errrel, dOne
      integer i, maxfn, num, iso, uniaxl, biaxl
      parameter ( delR=0.1, delta=1.0e-6, num=10 )
      parameter ( errabs=0.0, errrel=1.0e-5, dOne=1.0 )
      parameter ( iso = 1, uniaxl = 2, biaxl = 3 )
C
      real r(num+1), er(num+1), a, b, newSgn, oldSgn
      integer count
C
      real e(3), alf(3), ec, kappa, Ka
      integer mu, type
      common kappa, Ka, e, ec, alf, mu, type
C
      real g1, g2
      external g1, g2
C
C.............................................................................
C
C     Calculate values of function between 0 and 1
C
      do 10 i = 1, num+1
        r(i) = (i-1)*delR
        if ( i .eq. 1 ) then
            r(i) = delta
          endif
        if ( select .eq. 1 ) then
            er(i) = g1( r(i) )
          else
            er(i) = g2( r(i) )
          endif
 10   continue
C
C     Look for sign change and then use library routine to find root
C
      count = 0
      oldSgn = sign( dOne, er(1) )
      do 20 i = 2, num+1
        newSgn = sign( dOne, er(i) )
        if ( newSgn .ne. oldSgn ) then
            count = count + 1
            a = r(i-1)
            b = r(i)
            maxfn = 15
```

```
            if ( select .eq. 1 ) then
                call zbren( g1, errabs, errrel, a, b, maxfn )
              else
                call zbren( g2, errabs, errrel, a, b, maxfn )
              endif
            if ( count .eq. 1 ) then
                r1 = b
              else
                r2 = b
              endif
          endif
        oldSgn = newSgn
 20   continue
C
      if ( count .eq. 2 ) then
          iroots = .true.
        else
          r1 = 0.0
          r2 = 0.0
          iroots = .false.
        endif
      return
      end
C
C***********************************************************************
C
      logical function uroots( r1, r2 )
C
C***********************************************************************
C
      real r1, r2
C
      real errabs, errrel, dOne
      integer maxfn, num, iso, uniaxl, biaxl
      parameter ( num=10 )
      parameter ( errabs=0.0, errrel=1.0e-5, dOne=1.0 )
      parameter ( iso = 1, uniaxl = 2, biaxl = 3 )
C
      real r(num+1), er(num+1), Rmax, delR, delta, a, b, newSgn, oldSgn
      integer count, i
C
      real e(3), alf(3), ec, kappa, Ka
      integer mu, type
      common kappa, Ka, e, ec, alf, mu, type
C
      real g1
      external g1
C
C.......................................................................
C
C     Calculate values of function between 0 and 1
C
      delR = 0.1
      if ( type .eq. biaxl ) then
          Rmax = ((e(2)-kappa**2)/(e(2)-ec))**(1.0/alf(2))
          delR = (Rmax-1.0e-12)/num
        endif
        delta = 1.0e-5 * delR
      do 10 i = 1, num+1
        r(i) = (i-1)*delR
        if ( i .eq. 1 ) then
            r(i) = delta
          endif
        er(i) = g1( r(i) )
 10   continue
C
C     Look for sign change and then use library routine to find root
C
      count = 0
```

```
      oldSgn = sign( dOne, er(1) )
      do 20 i = 2, num+1
        newSgn = sign( dOne, er(i) )
        if ( newSgn .ne. oldSgn ) then
            count = count + 1
            a = r(i-1)
            b = r(i)
            maxfn = 15
            call zbren( g1, errabs, errrel, a, b, maxfn )
            if ( count .eq. 1 ) r1 = b
            if ( count .eq. 2 ) r2 = b
          endif
        oldSgn = newSgn
20    continue
C
      if ( count .eq. 2 ) then
          uroots = .true.
        else
          if ( (count.eq.1)
     &          .and.(type.eq.biaxl).and.(alf(2).gt.alf(1)) ) then
              uroots = .true.
              r2 = Rmax
            else
              r1 = 0.0
              r2 = 0.0
              uroots = .false.
            endif
        endif
      return
      end
C
C**************************************************************************
C
      function psi1( r )
C
C**************************************************************************
C
      real r
C
      integer iso, uniaxl, biaxl
      parameter ( iso = 1, uniaxl = 2, biaxl = 3 )
C
      real e(3), alf(3), ec, kappa, Ka
      integer mu, type
      common kappa, Ka, e, ec, alf, mu, type
C
      real er1, er2, er3, x, KppSqr, term2
C
C..........................................................................
C
      KppSqr = kappa**2
      if ( type .eq. iso ) then
          x = e(1) - ( e(1) - ec )*abs(r)**alf(1) - KppSqr
        else
          er1 = e(1) - ( e(1) - ec )*abs(r)**alf(1)
          er2 = e(2) - ( e(2) - ec )*abs(r)**alf(2)
          er3 = e(3) - ( e(3) - ec )*abs(r)**alf(3)
          x = er3 - er3*(KppSqr)/er1
        endif
      if ( mu .ne. 0 ) then
          term2 = (mu/(Ka*abs(r)))**2
          if ( type .eq. biaxl ) then
              term2 = term2*er2*(er1-KppSqr)/(er1*(er2-KppSqr))
            endif
          x = x - term2
        endif
      if ( x .ge. 0.0 ) then
          psi1 = Ka*sqrt( x )
        else
```

```
          psi1 = 0.0
        endif
      return
      end
C
C**********************************************************************
C
      function psi2( r )
C
C**********************************************************************
C
      real r
C
      integer iso, uniaxl, biaxl
      parameter ( iso = 1, uniaxl = 2, biaxl = 3 )
C
      real e(3), alf(3), kappa, Ka
      integer mu, type
      common kappa, Ka, e, ec, alf, mu, type
C
      real er2, x
C
C.....................................................................
C
      er2 = e(2) - ( e(2) - ec )*abs(r)**alf(2)
      x = er2 - kappa**2
      if ( mu .ne. 0 ) then
          x =  x - (mu/(Ka*r))**2
        endif
      if ( x .ge. 0.0 ) then
          psi2 = Ka*sqrt( x )
        else
          psi2 = 0.0
        endif
      return
      end
C
C**********************************************************************
C
      function g1( r )
C
C**********************************************************************
C
      real r
C
      integer iso, uniaxl, biaxl
      parameter ( iso = 1, uniaxl = 2, biaxl = 3 )
C
      real e(3), alf(3), ec, kappa, Ka
      integer mu, type
      common kappa, Ka, e, ec, alf, mu, type
C
      real er1, er2, er3, KppSqr, term1, term2
C
C.....................................................................
C
      er1 = e(1) - ( e(1) - ec )*abs(r)**alf(1)
      er2 = e(2) - ( e(1) - ec )*abs(r)**alf(2)
      er3 = e(3) - ( e(3) - ec )*abs(r)**alf(3)
      KppSqr = kappa**2
      term1 = er1 - KppSqr
      term2 = ( mu / (Ka*abs(r)) )**2
      if (type .ne. iso ) term1 = er3*term1/er1
      if ( type .eq. biaxl ) then
          term2 = term2*er2*(er1-KppSqr)/(er1*(er2-KppSqr))
        endif
      g1 = term1 - term2
      return
      end
```

```
C
C***********************************************************************
C
      function g2( r )
C
C***********************************************************************
C
      real r
C
      integer iso, uniaxl, biaxl
      parameter ( iso = 1, uniaxl = 2, biaxl = 3 )
C
      real e(3), alf(3), ec, kappa, Ka
      integer mu, type
      common kappa, Ka, e, ec, alf, mu, type
C
C.......................................................................
C
      g2 = e(2) - ( e(2) -ec )*abs(r)**alf(2) - kappa**2
     &          - (mu/(Ka*abs(r)))**2
      return
      end
```

```
program Asymp( output );

  const
    maxSolnOrder = 10;
    maxSize = 100;
    mMaxPlus2 = 6;

  type
    integer2 = -32768..32767;
    complex = record
                x, y : real;
              end;
    modeType = ( HE, EH );
    matrix = array[1..4,1..4] of complex;
    superMatrix = array[0..maxSolnOrder] of matrix;
    eigenvalues = array[1..4] of real;
    vector = array[1..mMaxPlus2] of real;

  var
    noRootsFound : boolean;
    i, j, l, n, m, solnOrder, numKa, numKappa, oldSign,
      newSign, debug, colNum : integer2;
    n1, e1, n3, e3, nc, ec, Ka, KaMin, KaMax, kappaMin, kappaMax,
      delKappa, delKa, lowerKappa, upperKappa, lowerDet,
      upperDet, det, root, oldRoot : real;
    cmplxDeterminant : complex;
    kappa, determinant : array[1..maxSize] of real;
    data : text;

  procedure cmplx( a, b : real; var z : complex );
    begin { cmplx }
      z.x := a;
      z.y := b;
    end; { cmplx }

  procedure cmplxAdd( z1, z2 : complex; var result : complex );
    begin { cmplxAdd }
      result.x := z1.x + z2.x;
      result.y := z1.y + z2.y;
    end; { cmplxAdd }

  procedure cmplxSub( z1, z2 : complex; var result : complex );
    begin { cmplxSub }
      result.x := z1.x - z2.x;
      result.y := z1.y - z2.y;
    end; { cmplxSub }

  procedure cmplxMult( z1, z2 : complex; var result : complex );
    var
      temp : complex;
    begin { cmplxMult }
      temp.x := z1.x*z2.x - z1.y*z2.y;
      temp.y := z1.x*z2.y + z2.x*z1.y;
      result := temp;
    end; { cmplxMult }

  procedure scalarMult( a: real; z : complex; var result : complex );
    begin { scalarMult }
      result.x := a*z.x;
      result.y := a*z.y;
    end; { scalarMult }

  function RealPart( z : complex ) : real;
    begin { RealPart }
      RealPart := z.x;
    end; { RealPart }
```

```
procedure matAdd( A, B : matrix; var C : matrix );
  var
    i, j : integer2;
  begin { matAdd }
    for i := 1 to 4 do
      for j := 1 to 4 do
        cmplxAdd( A[i,j], B[i,j], C[i,j] );
  end; { matAdd }

procedure matSub( A, B : matrix; var C : matrix );
  var
    i, j : integer2;
  begin { matSub }
    for i := 1 to 4 do
      for j := 1 to 4 do
        cmplxSub( A[i,j], B[i,j], C[i,j] );
  end; { matSub }

procedure matMult( A, B : matrix; var C : matrix );
  var
    i, j, k : integer2;
    sum, product : complex;
  begin { matMult }
    for i := 1 to 4 do
      for j := 1 to 4 do
        begin
          cmplx( 0.0, 0.0, sum );
          for k := 1 to 4 do
            begin
              cmplxMult( A[i,k], B[k,j], product );
              cmplxAdd( sum, product, sum );
            end;
          C[i,j] := sum;
        end;
  end; { matMult }

procedure cmplxDet( A : matrix; var determinant : complex );
  type
    mat2 = array[1..2,1..2] of complex;
    mat3 = array[1..3,1..3] of complex;

  var
    subDet, sum, product : complex;
    subMat : mat3;
    i, j, k, linePnt : integer2;

  procedure cmplxDet2( A : mat2; var determinant : complex );
    var
      product1, product2 : complex;
    begin { cmplxDet2 }
      cmplxMult( A[1,1], A[2,2], product1 );
      cmplxMult( A[1,2], a[2,1], product2 );
      cmplxSub( product1, product2, determinant );
    end; { cmplxDet2 }

  procedure cmplxDet3( A : mat3; var determinant : complex );
    var
      subDet, sum, product : complex;
      subMat : mat2;
      i, j, k, linePnt : integer2;
    begin { cmplxDet3 }
      cmplx( 0.0, 0.0, sum );
      for i := 1 to 3 do
        begin
          linePnt := 1;
          for j := 1 to 3 do
            if ( j <> i ) then
```

```
              begin
                for k := 1 to 2 do
                  subMat[linePnt,k] := A[j,K+1];
                linePnt := linePnt + 1;
              end;
          cmplxDet2( subMat, subDet );
          cmplxMult( A[i,1], subDet, product );
          if odd(i)
            then cmplxAdd( sum, product, sum )
            else cmplxSub( sum, product, sum );
        end;
      determinant := sum;
    end; { cmplxDet3 }

  begin { cmplxDet }
    cmplx( 0.0, 0.0, sum );
    for i := 1 to 4 do
      begin
        linePnt := 1;
        for j := 1 to 4 do
          if ( j <> i ) then
            begin
              for k := 1 to 3 do
                subMat[linePnt,k] := A[j,K+1];
              linePnt := linePnt + 1;
            end;
        cmplxDet3( subMat, subDet );
        cmplxMult( A[i,1], subDet, product );
        if odd(i)
          then cmplxAdd( sum, product, sum )
          else cmplxSub( sum, product, sum );
      end;
    determinant := sum;
  end; { cmplxDet }

procedure IdentMat( var A : matrix );
  var
    cmplxZero, cmplxOne : complex;
    i, j : integer2;
  begin { IdentMat }
    cmplx( 0.0, 0.0, cmplxZero );
    cmplx( 1.0, 0.0, cmplxOne );
    for i := 1 to 4 do
      begin
        for j := 1 to 4 do
          A[i,j] := cmplxZero;
        A[i,i] := cmplxOne;
      end;
  end; { IdentMat }

procedure ZeroMat( var A : matrix );
  var
    cmplxZero : complex;
    i, j : integer2;
  begin { ZeroMat }
    cmplx( 0.0, 0.0, cmplxZero );
    for i := 1 to 4 do
      for j := 1 to 4 do
        A[i,j] := cmplxZero;
  end ; { ZeroMat }

function power( x : real; n : integer ) : real;
  var
    i : integer2;
    product : real;
  begin { power }
    product := x;
```

```
    for i := 1 to n-1 do
      product := x * product;
    power := product;
  end ; { power }
  { ------------}

procedure findP0 (m: integer; er, kappa, Ka: real; var P0: matrix);
  var
   fac1, fac2, fac3: real;

  begin { findP0 }
    ZeroMat(P0);
    fac1 := m * kappa;
    fac2 := m * er;
    fac3 := er - kappa * kappa;

    P0[1, 1].x := fac3;
    P0[1, 3].x := fac3;

    P0[2, 1].x := fac1;
    P0[2, 2].y := m;
    P0[2, 3].x := fac1;
    P0[2, 4].y := -1.0 * m;

    P0[3, 2].x := fac3;
    P0[3, 4].x := fac3;

    P0[4, 1].y := -fac2;
    P0[4, 2].x := fac1;
    P0[4, 3].y := fac2;
    P0[4, 4].x := fac1;
   end; { findP0 }

procedure findP0inv (m: integer;
                     er, kappa, Ka: real;
                     var P0inv: matrix );
  var
   fac0, fac1, fac2, fac3, fac4, fac5: real;

  begin { findP0inv }
    ZeroMat(P0inv);
    fac1 := 1.0 / (2 * (er - kappa * kappa));
    fac2 := kappa * fac1;
    fac3 := fac2 / er;
    fac4 := 1.0 / (2 * m);
    fac5 := fac4 / er;

    P0inv[1, 1].x := fac1;
    P0inv[1, 3].y := -1.0 * fac3;
    P0inv[1, 4].y := fac5;

    P0inv[2, 1].y := fac2;
    P0inv[2, 2].y := -1.0 * fac4;
    P0inv[2, 3].x := fac1;

    P0inv[3, 1].x := fac1;
    P0inv[3, 3].y := fac3;
    P0inv[3, 4].y := -1.0 * fac5;

    P0inv[4, 1].y := -1.0 * fac2;
    P0inv[4, 2].y := fac4;
    P0inv[4, 3].x := fac1;
  end; { findP0inv }

{ The implementation of findAn depends upon the type of fiber }
{ for which the problem is being solved. Separate version  of }
{ findAn for a uniaxial step-index fiber, a uniaxial          }
```

```
{ parabolic-index fiber and a biaxial graded-index fiber can  }
{ be found following this program listing.                    }

procedure findAn( n, m : integer2;
                  e1, e3, ec, kappa, Ka : real;
                  var An : matrix );

  begin { findAn }
  end ; { findAn }

procedure findBn( Fn : matrix; var Bn : matrix );
  var
    i : integer2;
  begin { findBn }
    ZeroMat( Bn );
    for i := 1 to 4 do
      Bn[i,i] := Fn[i,i];
  end; { findBn }

procedure findWn( Fn : matrix; n : integer2;
                  evals : eigenvalues; typeOfMode : modeType;
                  var Wn : matrix );
  var
    i, j, jLow, jHigh : integer2;
    cmplxZero : complex;
  begin { findWn }
    ZeroMat( Wn );
    jLow := 1;
    jHigh := 2;
    if ( typeOfMode = EH ) then
      begin
        jLow := 3;
        jHigh := 4;
      end;
    for i := 1 to 4 do
      for j := jLow to jHigh do
        if ( i <> j )
          then scalarMult( -1.0/(evals[i]-evals[j]-n), Fn[i,j],
                                 Wn[i,j]);
  end; { findWn }

procedure findFn( P0inv : matrix;
                  var  A, B, P : superMatrix;
                  n : integer2; var Fn : matrix );
  var
    term1, product1, product2, sum : matrix;
    l : integer2;
  begin { findFn }
    matMult( A[n], P[0], term1 );
    if ( n > 1 ) then
      begin
        ZeroMat( sum );
        for l := 1 to n-1 do
          begin
            matMult( A[n-l], P[l], product1 );
            matMult( P[l], B[n-l], product2 );
            matAdd( sum, product1, sum );
            matSub( sum, product2, sum );
          end;
      end;
    matAdd( term1, sum, sum );
    matMult( P0inv, sum, Fn );
  end; { findFn }

function dbsk0( var x : real ) : real;
  fortran;
```

```
function dbsk1( var x : real ) : real;
  fortran;

procedure dbsks( var xnu, x : real;
                 var n : integer2;
                 var bsk : vector );
  fortran;

procedure vsFort; fortran;

procedure WriteMat( A : matrix );
  var
    i, j : integer2;
  begin { WriteMat }
    for i := 1 to 4 do
      begin
        for j := 1 to 4 do
          write( '(', A[i,j].x, ',', A[i,j].y, ') ' );
        writeln;
      end;
  end; { WriteMat }

procedure findDet( solnOrder, m : integer2;
                   e1, e2, e3, kappa, Ka : real;
                   var determinant : real );
  var
    typeOfMode : modeType;
    i, n, r, c, col, cLow, cHigh, delC, size, absM : integer2;
    powerOfS, KaSqr, gmmNSq, gmmN, x, Km, KmPrime, xnu : real;
    cmplxDeterminant : complex;
    Vi, bsk : vector;
    evals : eigenvalues;
    temp : complex;
    A, B, F, P, W : superMatrix;
    P0inv, zero, approxP, bndCon : matrix;

  begin { findDet }
    if ( m > 0 )
      then typeOfMode := HE
      else typeOfMode := EH;

    ZeroMat( zero );
    for i := 0 to solnOrder do
      begin
        A[i] := zero;
        B[i] := zero;
        F[i] := zero;
        P[i] := zero;
        W[i] := zero;
      end;
    evals[1] := m;
    evals[2] := m;
    evals[3] := -1*m;
    evals[4] := -1*m;
    for i := 1 to 4 do
      cmplx( evals[1], 0.0, B[0,i,i] );

    findP0( m, e1, kappa, Ka, P[0] );
    findP0inv( m, e1, kappa, Ka, P0inv );
    findAn( 0, m, e1, e3, ec, kappa, Ka, A[0] );

    if ( solnOrder > 1 ) then
      for n := 2 to solnOrder do
        begin
          findAn( n, m, e1, e3, ec, kappa, Ka, A[n] );
          findFn( P0inv, A, B, P, n, F[n] );
          findBn( F[n], B[n] );
          findWn( F[n], n, evals, typeOfMode, W[n] );
```

```
        matMult( P[0], W[n], P[n] );
      end;

    { Find boundary condition matrix }

    powerOfS := 1;
    KaSqr := Ka * Ka;
    approxP := zero;
    cLow := 1;
    cHigh := 2;
    delC := 0;
    if ( typeOfMode = EH ) then
      begin
        cLow := 3;
        cHigh := 4;
        delC := -2;
      end;

    for n := 0 to solnOrder do
      if not odd(n) then
        begin
          for r := 1 to 4 do
            for c := cLow to cHigh do
              begin
                scalarMult( powerOfS, P[n,r,c], temp);
                CmplxAdd( approxP[r,c],
                            temp,approxP[r,c] );
              end;
          if ( n = 0 ) then
            begin
              Vi[1] := power( Ka, abs(m) );
              Vi[2] := Vi[1];
            end
          else
            begin
              Vi[1] := Vi[1] * exp( RealPart( B[n,cLow,cLow] )
                                   * powerOfS / n );
              Vi[2] := Vi[2] * exp( RealPart( B[n,cHigh,cHigh] )
                                   * powerOfS / n );
            end;
          powerOfS := KaSqr * powerOfS;
        end;

    bndCon := zero;
    for r := 1 to 4 do
      for c := cLow to cHigh do
        begin
          col := c + delC;
          scalarMult( Vi[col], approxP[r,c], bndCon[r,col] );
          if not odd( r ) then
            scalarMult( 1.0/Ka, bndCon[r,c], bndCon[r,col] );
        end;

    gmmNSq := kappa * kappa - ec;
    gmmN := sqrt( gmmNSq );
    x := Ka * gmmN;
    if ( m = 0 ) then
      begin
        Km := dbsk0( x );
        KmPrime := -1.0 * dbsk1( x );
      end
    else
      begin
        absM := abs( m );
        size := absM + 2;
        xnu := 0.0;
        dbsks( xnu, x, size, bsk );
        Km := bsk[absM+1];
```

```
          KmPrime := -0.5*( bsk[absM] + bsk[absM+2] );
        end;
      Cmplx( -1.0*Km, 0.0, bndCon[1,3] );
      bndCon[3,4] := bndCon[1,3];
      Cmplx( m*kappa*Km/(Ka*gmmNSq), 0.0, bndCon[2,3] );
      bndCon[4,4] := bndCon[2,3];
      Cmplx( 0.0, KmPrime/gmmN, bndCon[2,4] );
      Cmplx( 0.0, -1.0*ec*KmPrime/gmmN, bndCon[4,3] );

      CmplxDet( bndCon, cmplxDeterminant );
      determinant := cmplxDeterminant.x;
    end; { findDet }

  function sign( x : real ) : integer2;
    begin { sign }
      if ( x < 0.0 )
        then sign := -1
        else sign := 1;
    end; { sign }

  function findRoot( x1, y1, x2, y2 : real ) : real;
    var
      slope : real;
    begin { findRoot }
      slope := ( y2-y1 )/( x2-x1 );
      findRoot := x1 - y1/slope;
    end; { findRoot }

  begin { Asymp }
    vsFort;

    reset( data );
    readln( data, m, solnOrder );
    readln( data, n1, n3, nc );
    readln( data, KaMin, KaMax, numKa );
    readln( data, kappaMin, kappaMax, numKappa );
    readln( data, debug, colNum );

    e1 := n1*n1;
    e3 := n3*n3;
    ec := nc*nc;

    delKa := ( KaMax-KaMin )/numKa;
    delKappa := ( kappaMax-kappaMin )/numKappa;

    Ka := KaMin;
    for i:= 1 to numKa do
      begin
        for j := 1 to numKappa do
          begin
            kappa[j] := kappaMax - (j-1)*delKappa;
            findDet( solnOrder, m, e1, e1, e3,
                     kappa[j], Ka, determinant[j] );
            if ( debug = 0 ) then
              writeln( Ka:5:2, kappa[j]:11:6, determinant[j] );
          end; { for j }

        { Find roots by looking for changes in sign of determinant }

        oldSign := sign( determinant[1] );
        j := 1;
        noRootsFound := true;
        while ( noRootsFound and ( j <= numKappa ) ) do
          begin { while }
            j := j + 1;
            newSign := sign( determinant[j] );
```

```
          if ( newSign <> oldSign )
            then
              begin
                noRootsFound := false;
                lowerKappa := kappa[j];
                lowerDet := determinant[j];
                upperKappa := kappa[j-1];
                upperDet := determinant[j-1];
                oldRoot := 1.49;
                root := lowerKappa;
                while ( abs( oldRoot-root ) > 1.0e-6 ) do
                  begin
                    oldRoot := root;
                    root := findRoot( lowerKappa, lowerDet,
                                      upperKappa, upperDet );
                    findDet( solnOrder, m, e1, e1, e3,
                             root, Ka, det );
                    if ( det < 0.0 )
                      then
                        begin
                          lowerKappa := root;
                          lowerDet := det;
                        end
                      else
                        begin
                          upperKappa := root;
                          upperDet := det;
                        end;
                  end; { while }
              end;
          oldSign := newSign;
        end; { while }
      if noRootsFound
        then writeln( Ka:4:1,'      no roots' )
        else writeln( Ka:4:1, '<', colNum:1,'> ', root:10:8);

      Ka := Ka + delKa
    end; { for i }
end. { Asymp }

{ findAn for a step-index fiber }

procedure findAn( n, m : integer;
                  e1, e3, ec, kappa, Ka : real;
                  var An : matrix );

  var
    fac1, fac2, fac3, fac4, fac5, fac6, fac7, del : real;

  begin { findAn }
    ZeroMat( An );
    if ( n = 0 )then
      begin
        fac1 := m * kappa;
        fac2 := kappa * kappa;
        fac3 := e1 - fac2;
        fac4 := m * m;
        An[1,3].y := -1.0*fac1/e1;
        An[1,4].y := fac3/e1;
        An[2,3].y := fac4/e1;
        An[2,4].y := fac1/e1;
        An[3,1].y := fac1;
        An[3,2].y := -1.0*fac3;
        An[4,1].y := -1.0*fac4;
        An[4,2].y := -1.0*fac1;
      end;
```

```
  if ( n = 2 ) then
   begin
    An[2,3].y := -1;
    An[4,1].y := e3;
   end;

 end ; { findAn }

{ findAn for an uniaxial graded-index fiber }
{ where e1 and e3 have parabolic profiles   }

procedure findAn( n, m : integer;
         e1, e3, ec, kappa, Ka : real;
         var An : matrix );

 var
  fac1, fac2, fac3, fac4, fac5, fac6, fac7, del : real;

 begin { findAn }
  ZeroMat( An );
  if not odd(n) then
   begin
    fac1 := m * kappa;
    fac2 := kappa * kappa;
    fac3 := e1 - fac2;
    fac4 := m * m;
    if ( n = 0 ) then
      begin
       An[1,3].y := -1.0*fac1/e1;
       An[1,4].y := fac3/e1;
       An[2,3].y := fac4/e1;
       An[2,4].y := fac1/e1;
       An[3,1].y := fac1;
       An[3,2].y := -1.0*fac3;
       An[4,1].y := -1.0*fac4;
       An[4,2].y := -1.0*fac1;
      end
     else
      begin
       del := ( e1 - ec ) / ( 2 * e1 );
       fac5 := 2 * del / ( Ka * Ka );
       fac6 := power( fac5, n div 2 );
       fac7 := fac1*fac6/e1;
       An[1,3].y := -1.0*fac7/e1;
       An[2,4].y := fac7;
       An[1,4].y := -1.0*fac2*fac6/e1;
       if ( n = 2 ) or ( n = 4 ) then
        case n of
         2 : begin
            An[2,3].y := fac4*fac6/e1 - 1.0;
            An[3,2].y := e1*fac6;
            An[4,1].y := e3;
           end;
         4 : begin
            An[2,3].y := fac4*fac6/e1;
            An[4,1].y := ( ec - e3 )/( Ka * Ka );
           end;
        end
       else An[2,3].y := fac4*fac6/e1;
      end;
   end;
 end ; { findAn }

{ findAn for a biaxial graded-index fiber  }
{ where e1 and e3 have a parabolic profile }
{ and e2 is constant                       }
```

```
procedure findAn( n, m : integer2;
                  e1, e3, ec, kappa, Ka : real;
                  var An : matrix );

  var
    fac1, fac2, fac3, fac4, fac5, fac6, fac7, del : real;

  begin { findAn }
    ZeroMat( An );
    if not odd(n) then
      begin
        fac1 := m * kappa;
        fac2 := kappa * kappa;
        fac3 := e1 - fac2;
        fac4 := m * m;
        if ( n = 0 ) then
            begin
              An[1,3].y := -1.0*fac1/e1;
              An[1,4].y := fac3/e1;
              An[2,3].y := fac4/e1;
              An[2,4].y := fac1/e1;
              An[3,1].y := fac1;
              An[3,2].y := -1.0*fac3;
              An[4,1].y := -1.0*fac4;
              An[4,2].y := -1.0*fac1;
            end
          else
            begin
              del := ( e1 - ec ) / ( 2 * e1 );
              fac5 := 2 * del / ( Ka * Ka );
              fac6 := power( fac5, n div 2 );
              fac7 := fac1*fac6/e1;
              An[1,3].y := -1.0*fac7/e1;
              An[2,4].y := fac7;
              An[1,4].y := -1.0*fac2*fac6/e1;
              if ( n = 2 ) or ( n = 4 ) then
                case n of
                  2 : begin
                        An[2,3].y := fac4*fac6/e1 - 1.0;
                        An[4,1].y := e3;
                      end;
                  4 : begin
                        An[2,3].y := fac4*fac6/e1;
                        An[4,1].y := ( ec - e3 )/( Ka * Ka );
                      end;
                end
              else An[2,3].y := fac4*fac6/e1;
            end;
      end;
  end ; { findAn }
```

```
C***********************************************************************
C
C This program uses an approximate analytical method to calculate
C the dispersion curves for an uniaxial graded index fiber.
C
C***********************************************************************
C
C     Global Constants
C
C       numMax = maximum number of layers
C       dSize = size of determinant array
C
      integer numMax, dSize
C
      parameter ( numMax=10, dSize=100 )
C
C     Input Parameters
C
C       n1 = maximum value of the refractive index of the core
C            in the rho and phi directions
C       n3 = maximum value of the refractive index in the z direction
C       nc = refractive index of the cladding
C       a = radius of the core
C       mu = mode order of the solution
C       alf1, alf3  = parameters which describes the shape of the
C                     refractive index profiles
C       num = number of layers ( num .le. numMax )
C       KaMin = minimum value of Ka
C       KaMax = maximum value of Ka
C       numKa = number of divisions between KaMax and KaMin
C       KppMax = maximum value of kappa
C       KppMin = minimum value of kappa
C       numKpp = number of divisions between KppMax and KppMin
C
      real*8 n1, n3, nc, alf1, alf3, KaMax, KaMin, KppMax, KppMin,
     &       delKa, delKpp
      integer mu, num, numKa, numKpp
C
C     Computed Parameters
C
C       e1 = maximum value of permittivity of the core in the rho and
C              phi directions
C            = n1**2
C       e3 = maximum value of permitivity of the core in the z
C              direction
C            = n3**2
C       ec = permittiviy of cladding = nc**2
C       del1 = ( e1-ec )/( 2*e1 )
C       del3 = ( e3-ec )/( 2*e3 )
C       delKa = increment for Ka = ( KaMax-KaMin )/numKa
C       delKpp = increment for kappa = ( KppMax-KppMin )/numKpp
C       rStep = increment for radius of layers = 1.0/num
C       rm = array containing radius for each layer
C       em1 = array containing permittivity in the rho and phi direction
C             for each layer
C       em3 = array containing permittivity in the z direction
C
      real*8 e1, e3, ec, del1, del3, Kastep, rstep, rm(numMax+1),
     &       em1(numMax+1), em3(numMax+1)
C
C     Program Variables
C
C       Ka = ko * a = normalized wave number
C
C       kappa = normalized propagation constant in the longitudinal
C               direction
C       i,j,k,loopKp,loopKa = loop variables
C
      real*8 Ka, kappa, K(dsize), D(dSize), nrMin,
```

```
     K      oldSgn, newSgn, oldDel, newDel
      integer i, loopKa, loopKp, loopB
C
C     Parameters for finding roots
C
      real*8 errabs, errrel, a, b, eps, eta, xguess(2), x(2)
      integer maxfn, nroots, itmax, infer(2)
      parameter ( errabs = 0.0, errrel = 1e-6, maxfn = 10 )
      parameter ( eps = 0.0, eta = 1e-5 )
      parameter ( nroots = 2, itmax = 10 )
C
C     Declarations needed to make findD a function of one variable
C     so that it can be used with dzbren and dzreal.
C
      real*8 findD
      external findD
      common em1, em3, rm, Ka, mu, num
C
C.....................................................................
C
C     Read input parameters
C
      read (7,*) num
      read (7,*) n1, alf1
      read (7,*) n3, alf3
      read (7,*) nc, mu
      read (7,*) KaMin,KaMax,numKa
      read (7,*) KppMin, KppMax, numKpp
C
      if ( num .gt. numMax ) num = numMax
      if ( numKpp .gt. dsize ) numKpp = dsize
C
      print *, 'n1 = ', n1, 'profile parameter = alf1 = ', alf1
      print *, 'n3 = ', n3, 'profile parameter = alf3 = ', alf3
      print *, 'nc = ', nc
      print *, 'number of layers = ', num
      print *, 'mu = ', mu
C
C     Calculate radius of each layer
C
      rstep = 1.0/num
      rm(1) = rstep
      if ( num .gt. 1 ) then
          do 10 i=2,num
            rm(i)=rm(i-1)+rstep
 10       continue
        endif
C
C     Calculate values of em
C
      e1 = n1**2
      e3 = n3**2
      ec = nc**2
      del1 = ( e1-ec )/( 2*e1 )
      del3 = ( e3-ec )/( 2*e3 )
C
      em1(1) = e1
      em3(1) = e3
      do 11 i = 2, num
        em1(i) = e1*( 1 - 2*del1*rm(i-1)**alf1 )
        em3(i) = e3*( 1 - 2*del3*rm(i-1)**alf3 )
 11   continue
      em1(num+1) = ec
      em3(num+1) = ec
C
      delKa = ( KaMax-KaMin )/numKa
      nrMin = dMin1( n1, n3 )
      if ( KppMax .gt. nrMin ) KppMax = nrMin
      delKpp = ( KppMax-KppMin )/numKpp
```

```
C
C     Loop through values of Ka
C
      Ka = KaMin
      do 70 loopKa = 1, numKa+1
C
C       Loop through values of B
C
        kappa = KppMax + delKpp
        do 40 loopKp = 1, numKpp
          kappa = kappa - delKpp
          K(loopKp) = kappa
          D(loopKp) = findD( kappa )
40      continue
C
C       Find roots by looking for a change in the
C       sign of the determinant
C
        oldSgn = dsign( 1.0d0, D(1) )
        do 50 i = 2, numKpp
          newSgn = dsign( 1.0d0, D(i) )
          if ( newSgn*oldSgn .lt. 0.0 ) then
             a = K(i)
             b = K(i-1)
             call dzbren( findD, errabs, errrel, a, b, maxfn )
             print 100, Ka, b
          endif
          oldSgn = newSgn
50      continue
C
C       Look for closely spaced roots
C
        oldDel = D(2) - D(1)
        oldSgn = dsign( 1.0d0, oldDel )
        do 60 i = 3, numKpp
          newDel = D(i) - D(i-1)
          newSgn = dsign( 1.0d0, newDel )
          if ( newSgn .ne. oldSgn ) then
              if ((D(i)*D(i-1) .gt. 0.0).and.(D(i)*newDel .gt. 0)) then
                  xguess(1) = K(i-1)
                  xguess(2) = K(i)
                  call dzreal( findD, errabs, errrel, eps, eta, nroots,
     &                             itmax, xguess, x, infer )
                  print 100, Ka, x(1)
                  print 100, Ka, x(2)
                endif
            endif
          oldDel = newDel
          oldSgn = newSgn
60      continue
C
        Ka = Ka + delKa
70    continue
C
C.....................................................................
C
100   format( 1X, '>>Root at Ka = ', F5.0, ' , kappa = ', F10.8 )
C
C.....................................................................
C
      stop
      end
C
C*********************************************************************
C
      subroutine ident( mat )
C
C*********************************************************************
```

```
C
      complex*16 mat(4,4)
      integer i, j
C
C.......................................................................
C
      do 92 i = 1, 4
        do 91 j = 1, 4
          mat(i,j) = ( 0.0, 0.0 )
          if ( i.eq.j ) then
              mat(i,j) = ( 1.0, 0.0 )
            endif
91      continue
92    continue
      return
      end
C
C***********************************************************************
C
      double precision function findD( kappa )
C
C***********************************************************************
C
      integer numMax
      parameter ( numMax = 10 )
C
      real*8 kappa
C
      real*8 em1(numMax+1), em3(numMax+1), rm(numMax+1), Ka
      integer mu, num
      common em1, em3, rm, Ka, mu, num
C
      complex*16 M1(4,4), Mm(4,4), Mtotal(4,4), MmInv(4,4),
     &           prod(4,4), bndcon(4,4)
      real*8 KppSq, pm2(numMax+1), D
      integer i, j, l, m
C
C.......................................................................
C
C                                                   (m)
C     Calculate values of transverse wave number p
C                                                   2
C
      KppSq = kappa**2
      do 15 i = 1, num+1
        pm2(i) = em1(i) - KppSq
15    continue
C
C     Find M (r )
C           1  1
C
      call findM( M1, rm(1), pm2(1), em1(1),
     &            em3(1), kappa, Ka, mu, 1, num )
C
C                        -1                        -1
C     Find product M (r )*M   (r )*...*M    (r     )*M    (r    )
C                   2  1   2    2       num  num-1   num  num
C
      call ident( Mtotal )
      if ( num .gt. 1 ) then
          do 20 m = 2, num
C
C           Find M (r   )
C                 m  m-1
C
            call findM( Mm, rm(m-1), pm2(m), em1(m),
     &                  em3(m), kappa, Ka, mu, m, num )
            call dmcrcr( 4, 4, Mtotal, 4, 4, 4, Mm, 4, 4, 4, prod, 4 )
            call dccgcg( 4, prod, 4, Mtotal, 4 )
```

```
C
C                 -1
C           Find M  (r )
C                 m   m
C
            call findMI( MmInv, rm(m), pm2(m), em1(m),
     &                   em3(m), kappa, Ka, mu, m, num )
            call dmcrcr( 4, 4, Mtotal, 4, 4, 4, MmInv, 4, 4, 4, prod,4 )
            call dccgcg( 4, prod, 4, Mtotal, 4 )
20        continue
        endif
C
C     Find M      (r    )
C           num+1   num
C
      l = num+1
      call findM( Mm, rm(num), pm2(l), em1(l),
     &              em3(l), kappa, Ka, mu, l, num )
      call dmcrcr( 4, 4, Mtotal, 4, 4, 4, Mm, 4, 4, 4, prod, 4)
      call dccgcg( 4, prod, 4, Mtotal, 4 )
C
C     Find overall matrix which combines all the boundary
C     conditions and find determinant
C
      do 32 i = 1, 4
        do 30 j = 1, 2
          bndcon(i,j) = M1(i,j)
30      continue
        do 31 j = 3,4
          bndcon(i,j) = -1.0*Mtotal(i,j)
31      continue
32    continue
      call det( bndcon, D )
      findD = D
      return
      end
C
C*************************************************************************
C
      subroutine findM( M, r, ktNsq, eps1, eps3, kappa, Ka, mu, layer,
     &                  num  )
C
C*************************************************************************
C
      complex*16 M(4,4)
      real*8 r, ktNSq, eps1, eps3, kappa, Ka
      integer mu, layer, num
C
      real*8 c1, c2, d1, d2, e1, e2, f1, f2, x, gmmN, ktN, k1,
     &       k2, zero, fac1, fac2
C
      data zero / 0.0 /
C
C.........................................................................
C
      fac1 = dsqrt( eps3/eps1 )
      fac2 = dsqrt( eps1*eps3 )
      if ( mu .eq. 0 ) then
          k1 = 0.0
        else
          k1 = mu*kappa/(Ka*ktNSq)
        endif
      if ( ktNSq .gt. 0.0 ) then
          ktN = sqrt( ktNSq )
          k2 = r/ ktN
          x = Ka*ktN*r
          call bessel( fac1*x, mu, c1, c2, d1, d2, layer )
          call bessel( x, mu, e1, e2, f1, f2, layer )
        else
```

```
          gmmN = sqrt( -1.0*ktNSq )
          k2 = -1.0*r / gmmN
          x = Ka*gmmN*r
          call mbessl( fac1*x, mu, c1, c2, d1, d2, layer, num )
          call mbessl( x, mu, e1, e2, f1, f2, layer, num )
        endif
C
      M(1,1) = dcmplx( c1 )
      M(1,2) = ( 0.0, 0.0 )
      M(1,3) = dcmplx( d1 )
      M(1,4) = ( 0.0, 0.0 )
C
      M(2,1) = ( 0.0, 0.0 )
      M(2,2) = dcmplx( e1 )
      M(2,3) = ( 0.0, 0.0 )
      M(2,4) = dcmplx( f1 )
C
      M(3,1) = dcmplx( k1*c1 )
      M(3,2) = dcmplx( zero, k2*e2 )
      M(3,3) = dcmplx( k1*d1 )
      M(3,4) = dcmplx( zero, k2*f2 )
C
      M(4,1) = dcmplx( zero, -1.0*k2*fac2*c2 )
      M(4,2) = dcmplx( k1*e1 )
      M(4,3) = dcmplx( zero, -1.0*k2*fac2*d2 )
      M(4,4) = dcmplx( k1*f1 )
C
      return
      end
C
C***********************************************************************
C
      subroutine findMI( M, r, ktNSq, eps1, eps3, kappa, Ka, mu, layer,
     &                   num  )
C
C***********************************************************************
C
      complex*16 M(4,4)
      real*8 r, ktNSq, eps1, eps2, kappa, Ka
      integer mu, layer, num
C
C.......................................................................
C
      call findM( M, r, ktNSq, eps1, eps3, kappa, Ka, mu, layer, num )
      call dlincg( 4, M, 4, M, 4 )
      return
      end
C
C***********************************************************************
C
      subroutine bessel( x, mu, c1, c2, d1, d2, layer )
C
C***********************************************************************
C
      real*8 x, c1, c2, d1, d2
      integer mu, layer
C
      real*8 xnu
      integer mumax
      parameter ( xnu = 0.0, mumax = 4 )
C
      real*8 bsj(mumax+2), bsy(mumax+2)
C
C.......................................................................
C
      if ( mu .gt. mumax ) then
           print *,'>>>Error: mu must be less than or equal to',mumax
           print *,'>>>       Program terminated due to error.'
```

```
          stop
        endif
      if ( layer .eq. 1 ) then
          d1 = 0.0
          d2 = 0.0
          if ( mu .eq. 0 ) then
              c1 = dbsj0( x )
              c2 = -1.0*dbsj1( x )
            else
              call dbsjs( xnu, x, mumax+2, bsj )
              c1 = bsj(mu+1)
              c2 = 0.5*( bsj(mu)-bsj(mu+2) )
            endif
        else
          if ( mu .eq. 0 ) then
              c1 = dbsj0( x )
              c2 = -1.0*dbsj1( x )
              d1 = dbsy0( x )
              d2 = -1.0*dbsy1( x )
            else
              call dbsjs( xnu, x, mumax+2, bsj )
              call dbsys( xnu, x, mumax+2, bsy )
              c1 = bsj(mu+1)
              c2 = 0.5*( bsj(mu)-bsj(mu+2) )
              d1 = bsy(mu+1)
              d2 = 0.5*( bsy(mu)-bsy(mu+2) )
            endif
        endif
      return
      end
C
C*********************************************************************
C
      subroutine mbessl( x, mu, c1, c2, d1, d2, layer, num )
C
C*********************************************************************
C
      real*8 x, c1, c2, d1, d2
      integer mu, layer, num
C
      real*8 xnu
      integer mumax
      parameter ( xnu = 0.0, mumax = 4 )
C
      real*8 bsi(mumax+2),bsk(mumax+2)
C
C.....................................................................
C
      if ( mu .gt. mumax ) then
          print *,'>>>Error: mu must be less than or equal to',mumax
          print *,'>>>       Program terminated due to error.'
          stop
        endif
      if ( layer .eq. 1 ) then
          d1 = 0.0
          d2 = 0.0
          if ( mu .eq. 0 ) then
              c1 = dbsi0( x )
              c2 = dbsi1( x )
            else
              call dbsis( xnu, x, mumax+2, bsi )
              c1 = bsi(mu+1)
              c2 = 0.5*( bsi(mu)+bsi(mu+2) )
            endif
        endif
      if (( layer .gt. 1 ) .and. ( layer .le. num )) then
          if ( mu .eq. 0 ) then
              c1 = dbsi0( x )
```

```
              c2 = dbsi1( x )
              d1 = dbsk0( x )
              d2 = -1.0*dbsk1( x )
             else
              call dbsis( xnu, x, mumax+2, bsi )
              call dbsks( xnu, x, mumax+2, bsk )
              c1 = bsi(mu+1)
              c2 = 0.5*( bsi(mu)+bsi(mu+2) )
              d1 = bsk(mu+1)
              d2 = -0.5*( bsk(mu)+bsk(mu+2) )
             endif
       endif
      if ( layer .eq. num+1 ) then
        c1 = 0.0
        c2 = 0.0
        if ( mu .eq. 0 ) then
          d1 = dbsk0( x )
          d2 = -1.0*dbsk1( x )
         else
          call dbsks( xnu, x, mumax+2, bsk )
          d1 = bsk(mu+1)
          d2 = -0.5*( bsk(mu)+bsk(mu+2) )
         endif
       endif
      return
      end
C
C************************************************************************
C
      subroutine det( mat, D )
C
C************************************************************************
C
      complex*16 mat(4,4)
      real*8 D
C
      complex*16 fac(4,4),det1
      real*8 det2
      integer n,lda,ldafac,ipvt(4)
C
      parameter ( n = 4, lda = 4, ldafac = 4 )
C
C.......................................................................
C
      call dlftcg( n, mat, lda, fac, ldafac, ipvt )
      call dlfdcg( n, fac, ldafac, ipvt, det1, det2 )
      D = dreal( det1 * 10.0**det2 )
      return
      end
```

Vita

Name:	Stephen F. Kawalko
Education:	B.S., University of Illinois at Chicago, 1987
Professional Memberships:	Institute of Electrical and Electronics Engineers IEEE Lasers and Electro-Optics Society
Abstracts:	Kawalko, S.F., and Uslenghi, P.L.E.: Guided propagation in graded-index anisotropic fibers. *National Radio Science Meeting Digest*, p 223, 1989